libstdc++

Generated by Doxygen 1.8.2

Fri Mar 1 2013 11:27:08

Contents

1 Todo List

Member __glibcxx_check_insert_range (_Position, _First, _Last)

We would like to be able to check for noninterference of _Position and the range [_First, _Last), but that can't (in general) be done.

Member __glibcxx_check_insert_range_after (_Position, _First, _Last)

We would like to be able to check for noninterference of _Position and the range [_First, _Last), but that can't (in general) be done.

Member __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_cxx::power (_Tp __x, _Integer __n)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::rope< _CharT, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Class __gnu_cxx::slist< _Tp, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> () const

Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

Member __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator-> () const

Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

Class std::basic_string< _CharT, _Traits, _Alloc >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::match_results< _Bi_iter, _Alloc >::format (_Out_iter __out, const char_type *__fmt_first, const char_type *__fmt_last, match_flag_type __flags=regex_constants::format_default) const

Implement this function.

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!= (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (int)

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ()

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (const regex_iterator &__rhs)

Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)

Implement this function.

Member std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Implement this function.

Implement this function.

Member std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)

Implement this function.

Member std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!= (const regex_token_iterator &__rhs)

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (int)

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (const regex_token_iterator &__rhs)

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (const regex_token_iterator &__rhs)

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector< int > &__submatches, regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ()

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[_Nm], regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (const regex_token_iterator &__rhs)

Implement this function.

Member std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default)

Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Member std::regex_traits< _Ch_type >::lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const

Implement this function.

Member std::regex_traits< _Ch_type >::lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const

Implement this function.

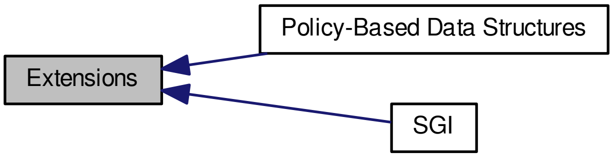
Member std::regex_traits< _Ch_type >::transform_primary (_Fwd_iter __first, _Fwd_iter __last) const

Implement this function.

2 Module Documentation

2.1 Extensions

Collaboration diagram for Extensions:



Modules

- Policy-Based Data Structures
- SGI

Classes

- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

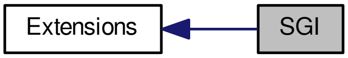
  Template class __versa_string.
  Data structure managing sequences of characters and character-like objects.

2.1.1 Detailed Description

Components generally useful that are not part of any standard.

2.2 SGI

Collaboration diagram for SGI:



Classes

- class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >

  An SGI extension .
- struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >

  An SGI extension .
- struct __gnu_cxx::constant_unary_fun< _Result, _Argument >

  An SGI extension .
- struct __gnu_cxx::constant_void_fun< _Result >

  An SGI extension .
- class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >
- struct __gnu_cxx::project1st< _Arg1, _Arg2 >

  An SGI extension .
- struct __gnu_cxx::project2nd< _Arg1, _Arg2 >

  An SGI extension .
- struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >
- class __gnu_cxx::rope< _CharT, _Alloc >
- struct __gnu_cxx::select1st< _Pair >

  An SGI extension .
- struct __gnu_cxx::select2nd< _Pair >

  An SGI extension .
- class __gnu_cxx::slist< _Tp, _Alloc >
- class __gnu_cxx::subtractive_rng
- struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >
- class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

  An SGI extension .

Functions

- template<typename _Tp >
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp , typename _Compare >
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- size_t std::_Find_first () const noexcept
- size_t std::_Find_next (size_t __prev) const noexcept
- template<class _Operation1 , class _Operation2 >
  unary_compose< _Operation1,
  _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1 , class _Operation2 , class _Operation3 >
  binary_compose< _Operation1,
  _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result >
  constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)
- template<class _Result >
  constant_unary_fun< _Result,
  _Result > __gnu_cxx::constant1 (const _Result &__val)
- template<class _Result >
  constant_binary_fun< _Result,
  _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  pair< _InputIterator,
  _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)
- template<typename _InputIterator , typename _Distance >
  void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<class _Tp >
  _Tp __gnu_cxx::identity_element (std::plus< _Tp >)
- template<class _Tp >
  _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)
- template<typename _InputIterator1 , typename _InputIterator2 >
  int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _Tp , typename _Integer , typename _MonoidOperation >
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _Tp , typename _Integer >
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n)
- template<typename _InputIterator , typename _RandomAccessIterator >
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator >
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator , typename _OutputIterator , typename _Distance >
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename _RandomNumberGenerator >
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)
- template<typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)

- bitset< _Nb > & std::_Unchecked_set (size_t __pos) noexcept
- bitset< _Nb > & std::_Unchecked_set (size_t __pos, int __val) noexcept
- bitset< _Nb > & std::_Unchecked_reset (size_t __pos) noexcept
- bitset< _Nb > & std::_Unchecked_flip (size_t __pos) noexcept
- constexpr bool std::_Unchecked_test (size_t __pos) const noexcept

2.2.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the `online documentation`, a copy of which is also shipped with the library source code (in .../docs/html/documentation.html). You can also read the documentation `on SGI's site`, which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f`(g(x)). The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f`(g1(x),g2(x)). The function compose2 takes f, g1, and g2, and constructs the `binary_compose` instance for you. For example, if `f` returns an int, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()s` take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()s` completely ignore any arguments passed, and return the stored value.

- `constant_void_fun's operator()` takes no arguments
- `constant_unary_fun's operator()` takes one argument (ignored)
- `constant_binary_fun's operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

2.2.2 Function Documentation

2.2.2.1 template<typename _Tp > const _Tp& __gnu_cxx::__median (const _Tp & *__a,* const _Tp & *__b,* const _Tp & *__c*)

Find the median of three values.

Parameters

__a	A value.
__b	A value.
__c	A value.

Returns

One of `a`, `b` or `c`.

If {`l`,m,n} is some convolution of {`a`,b,c} such that `l<=m<=n` then the value returned will be `m`. This is an SGI extension.

Definition at line 546 of file ext/algorithm.

2.2.2.2 template<typename _Tp , typename _Compare > const _Tp& __gnu_cxx::__median (const _Tp & *__a,* const _Tp & *__b,* const _Tp & *__c,* _Compare *__comp*)

Find the median of three values using a predicate for comparison.

Parameters

__a	A value.
__b	A value.
__c	A value.
__comp	A binary predicate.

Returns

One of `a`, `b` or `c`.

If {`l`,m,n} is some convolution of {`a`,b,c} such that `comp(l,m)` and `comp(m,n)` are both true then the value returned will be `m`. This is an SGI extension.

Definition at line 580 of file ext/algorithm.

2.2.2.3 size_t std::_Find_first ( ) const `[noexcept]`

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See Also

_Find_next

Definition at line 1354 of file bitset.

2.2.2.4 size_t std::_Find_next (size_t __prev) const [noexcept]

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

__prev	Where to start searching.

See Also

_Find_first

Definition at line 1365 of file bitset.

2.2.2.5 bitset<_Nb>& std::_Unchecked_flip (size_t __pos) [noexcept]

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1037 of file bitset.

Referenced by std::flip().

2.2.2.6 bitset<_Nb>& std::_Unchecked_reset (size_t __pos) [noexcept]

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1030 of file bitset.

Referenced by std::reset().

2.2.2.7 bitset<_Nb>& std::_Unchecked_set (size_t __pos) [noexcept]

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1013 of file bitset.

Referenced by std::set().

2.2.2.8 bitset<_Nb>& std::_Unchecked_set (size_t __pos, int __val) [noexcept]

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1020 of file bitset.

2.2.2.9 constexpr bool std::_Unchecked_test (size_t __pos) const [noexcept]

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1044 of file bitset.

Referenced by std::test().

2.2.2.10 template<class _Operation1 , class _Operation2 > unary_compose<_Operation1, _Operation2> __gnu_cxx::compose1 (const _Operation1 & __fn1, const _Operation2 & __fn2) [inline]

An SGI extension .

Definition at line 145 of file ext/functional.

2.2.2.11 template<class _Operation1 , class _Operation2 , class _Operation3 > binary_compose<_Operation1, _Operation2, _Operation3> __gnu_cxx::compose2 (const _Operation1 & *__fn1*, const _Operation2 & *__fn2*, const _Operation3 & *__fn3*) `[inline]`

An SGI extension .

Definition at line 172 of file ext/functional.

2.2.2.12 template<class _Result > constant_void_fun<_Result> __gnu_cxx::constant0 (const _Result & *__val*) `[inline]`

An SGI extension .

Definition at line 330 of file ext/functional.

2.2.2.13 template<class _Result > constant_unary_fun<_Result, _Result> __gnu_cxx::constant1 (const _Result & *__val*) `[inline]`

An SGI extension .

Definition at line 336 of file ext/functional.

2.2.2.14 template<class _Result > constant_binary_fun<_Result,_Result,_Result> __gnu_cxx::constant2 (const _Result & *__val*) `[inline]`

An SGI extension .

Definition at line 342 of file ext/functional.

2.2.2.15 template<typename _InputIterator , typename _Size , typename _OutputIterator > pair<_InputIterator, _OutputIterator> __gnu_cxx::copy_n (_InputIterator *__first*, _Size *__count*, _OutputIterator *__result*) `[inline]`

Copies the range [first,first+count) into [result,result+count).

Parameters

__first	An input iterator.
__count	The number of elements to copy.
__result	An output iterator.

Returns

A std::pair composed of first+count and result+count.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 120 of file ext/algorithm.

References std::__iterator_category().

2.2.2.16 template<typename _InputIterator , typename _Distance > void __gnu_cxx::distance (_InputIterator *__first*, _InputIterator *__last*, _Distance & *__n*) `[inline]`

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html`

Definition at line 105 of file ext/iterator.

References std::__iterator_category().

2.2.2.17 template<class _Tp > _Tp __gnu_cxx::identity_element (std::plus< _Tp >) [inline]

An SGI extension .

Definition at line 87 of file ext/functional.

2.2.2.18 template<class _Tp > _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >) [inline]

An SGI extension .

Definition at line 93 of file ext/functional.

2.2.2.19 template<typename _InputIterator1 , typename _InputIterator2 > int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 *__first1*, _InputIterator1 *__last1*, _InputIterator2 *__first2*, _InputIterator2 *__last2*)

`memcmp` on steroids.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.

Returns

An int, as with `memcmp`.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 201 of file ext/algorithm.

2.2.2.20 template<typename _Tp , typename _Integer , typename _MonoidOperation > _Tp __gnu_cxx::power (_Tp *__x*, _Integer *__n*, _MonoidOperation *__monoid_op*) [inline]

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 113 of file ext/numeric.

2.2.2.21 template<typename _Tp , typename _Integer > _Tp __gnu_cxx::power (_Tp *__x*, _Integer *__n*) [inline]

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 123 of file ext/numeric.

2.2.2.22 template<typename _InputIterator , typename _RandomAccessIterator > _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator *__first,* _InputIterator *__last,* _RandomAccessIterator *__out_first,* _RandomAccessIterator *__out_last*) `[inline]`

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 388 of file ext/algorithm.

2.2.2.23 template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator > _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator *__first,* _InputIterator *__last,* _RandomAccessIterator *__out_first,* _RandomAccessIterator *__out_last,* _RandomNumberGenerator & *__rand*) `[inline]`

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 411 of file ext/algorithm.

2.2.2.24 template<typename _ForwardIterator , typename _OutputIterator , typename _Distance > _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator *__first,* _ForwardIterator *__last,* _OutputIterator *__out,* const _Distance *__n*)

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 267 of file ext/algorithm.

References std::distance(), and std::min().

2.2.2.25 template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename _RandomNumberGenerator > _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator *__first,* _ForwardIterator *__last,* _OutputIterator *__out,* const _Distance *__n,* _RandomNumberGenerator & *__rand*)

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 301 of file ext/algorithm.

References std::distance(), and std::min().

2.2.2.26 template<typename _InputIter , typename _Size , typename _ForwardIter > pair<_InputIter, _ForwardIter> __gnu_cxx::uninitialized_copy_n (_InputIter *__first,* _Size *__count,* _ForwardIter *__result*) `[inline]`

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__count	Length
__result	An output iterator.

Returns

__result + (__first + __count)

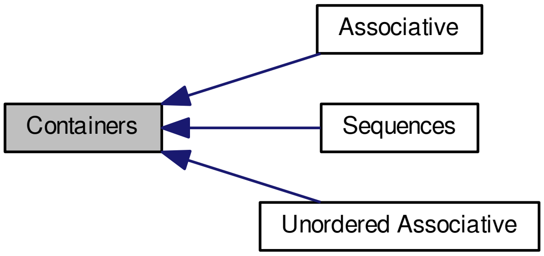
Like copy(), but does not require an initialized output range.

Definition at line 122 of file ext/memory.

References std::__iterator_category().

2.3 Containers

Collaboration diagram for Containers:



Modules

- Associative
- Sequences
- Unordered Associative

Classes

- class std::tr2::dynamic_bitset< _WordT, _Alloc >

  The dynamic_bitset class represents a sequence of bits.
- class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

2.3.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

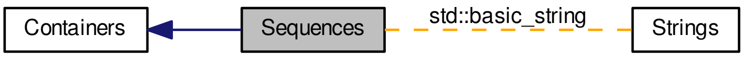
Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into Sequences and Associative Containers. Unordered Associative Containers.

2.4 Sequences

Collaboration diagram for Sequences:



Classes

- struct std::array< _Tp, _Nm >

    A standard container for storing a fixed size sequence of elements.
- class std::basic_string< _CharT, _Traits, _Alloc >

    Managing sequences of characters and character-like objects.
- class std::deque< _Tp, _Alloc >

    A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.
- class std::forward_list< _Tp, _Alloc >

    A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class std::list< _Tp, _Alloc >

    A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class std::priority_queue< _Tp, _Sequence, _Compare >

    A standard container automatically sorting its contents.
- class std::queue< _Tp, _Sequence >

    A standard container giving FIFO behavior.
- class std::stack< _Tp, _Sequence >

    A standard container giving FILO behavior.
- class std::vector< _Tp, _Alloc >

    A standard container which offers fixed time access to individual elements in any order.
- class std::vector< bool, _Alloc >

    A specialization of vector for booleans which offers fixed time access to individual elements in any order.

2.4.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

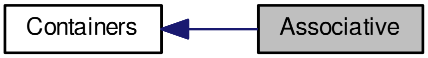
As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in `tables`.

2.5 Associative

Collaboration diagram for Associative:



Classes

- class std::map< _Key, _Tp, _Compare, _Alloc >

  A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class std::multimap< _Key, _Tp, _Compare, _Alloc >

  A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class std::multiset< _Key, _Compare, _Alloc >

  A standard container made up of elements, which can be retrieved in logarithmic time.
- class std::set< _Key, _Compare, _Alloc >

  A standard container made up of unique keys, which can be retrieved in logarithmic time.

2.5.1 Detailed Description

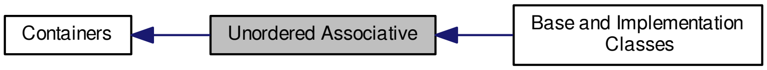
Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in `tables`.

2.6 Unordered Associative

Collaboration diagram for Unordered Associative:



Modules

- Base and Implementation Classes

Classes

- class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

  A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

  A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

  A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

  A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

2.6.1 Detailed Description

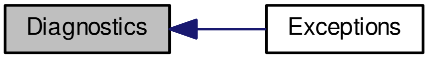
Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in `tables`.

2.7 Diagnostics

Collaboration diagram for Diagnostics:



Modules

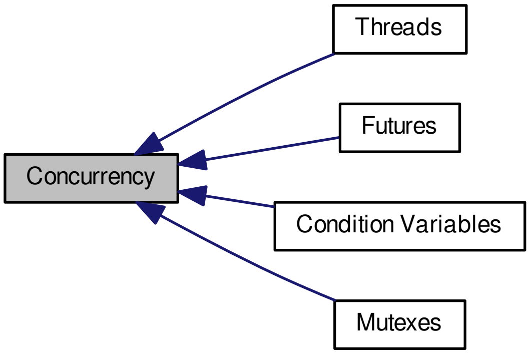
- Exceptions

2.7.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.8 Concurrency

Collaboration diagram for Concurrency:



Modules

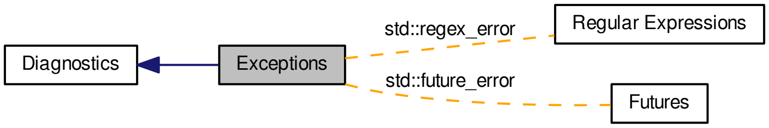
- Condition Variables
- Futures
- Mutexes
- Threads

2.8.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

2.9 Exceptions

Collaboration diagram for Exceptions:



Classes

- class __cxxabiv1::__forced_unwind

    Thrown as part of forced unwinding.
    A magic placeholder class that can be caught by reference to recognize forced unwinding.
- struct __gnu_cxx::forced_error

    Thown by exception safety machinery.
- class __gnu_cxx::recursive_init_error

    Exception thrown by __cxa_guard_acquire.
    6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.
- class std::__exception_ptr::exception_ptr

    An opaque pointer to an arbitrary exception.
- class std::bad_alloc

    Exception possibly thrown by `new`*.*
    `bad_alloc` *(or classes derived from it) is used to report allocation errors from the throwing forms of* `new`*.*
- class std::bad_cast

    Thrown during incorrect typecasting.
    If you attempt an invalid `dynamic_cast` *expression, an instance of this class (or something derived from this class) is thrown.*
- class std::bad_exception
- class std::bad_function_call

    Exception class thrown when class template function's operator() is called with an empty target.
- class std::bad_typeid

    Thrown when a NULL pointer in a `typeid` *expression is used.*
- class std::bad_weak_ptr

    Exception possibly thrown by `shared_ptr`*.*
- class std::domain_error
- class std::exception

    Base class for all library exceptions.
- class std::future_error

    Exception type thrown by futures.
- class std::invalid_argument
- class std::ios_base::failure

These are thrown to indicate problems with io.
27.4.2.1.1 Class ios_base::failure.

- class std::length_error
- class std::logic_error

  One of two subclasses of exception.
- class std::nested_exception

  Exception class with exception_ptr data member.
- class std::out_of_range
- class std::overflow_error
- class std::range_error
- class std::regex_error

  A regular expression exception class.
  The regular expression library throws objects of this class on error.
- class std::runtime_error

  One of two subclasses of exception.
- class std::system_error

  Thrown to indicate error code of underlying system.
- class std::underflow_error

Typedefs

- typedef void($*$ std::terminate_handler)()
- typedef void($*$ std::unexpected_handler)()

Functions

- template<typename _Ex >
  const nested_exception $*$ **std::__get_nested_exception** (const _Ex &__ex)
- template<typename _Ex >
  void **std::__throw_with_nested** (_Ex &&, const nested_exception $*$=0) __attribute__((__noreturn__))
- template<typename _Ex >
  void **std::__throw_with_nested** (_Ex &&,...) __attribute__((__noreturn__))
- void __gnu_cxx::__verbose_terminate_handler ()
- template<typename _Ex >
  exception_ptr std::copy_exception (_Ex __ex) noexcept
- exception_ptr std::current_exception () noexcept
- template<typename _Ex >
  exception_ptr std::make_exception_ptr (_Ex __ex) noexcept
- void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))
- template<typename _Ex >
  void std::rethrow_if_nested (const _Ex &__ex)
- void std::rethrow_if_nested (const nested_exception &__ex)
- terminate_handler std::set_terminate (terminate_handler) noexcept
- unexpected_handler std::set_unexpected (unexpected_handler) noexcept
- void std::terminate () noexcept __attribute__((__noreturn__))
- template<typename _Ex >
  void std::throw_with_nested (_Ex __ex)
- bool std::uncaught_exception () noexcept __attribute__((__pure__))
- void std::unexpected () __attribute__((__noreturn__))

2.9.1 Detailed Description

Classes and functions for reporting errors via exception classes.

2.9.2 Typedef Documentation

2.9.2.1 typedef void(∗ std::terminate_handler)()

If you write a replacement terminate handler, it must be of this type.

Definition at line 87 of file exception.

2.9.2.2 typedef void(∗ std::unexpected_handler)()

If you write a replacement unexpected handler, it must be of this type.

Definition at line 90 of file exception.

2.9.3 Function Documentation

2.9.3.1 void __gnu_cxx::__verbose_terminate_handler ( )

A replacement for the standard terminate_handler which prints more information about the terminating exception (if any) on stderr.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler
      )
```

to use. For more info, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.html`

In 3.4 and later, this is on by default.

2.9.3.2 template<typename _Ex > exception_ptr std::copy_exception (_Ex *__ex*) [noexcept]

Obtain an exception_ptr pointing to a copy of the supplied object.

Definition at line 167 of file exception_ptr.h.

References std::current_exception().

2.9.3.3 exception_ptr std::current_exception ( ) [noexcept]

Obtain an exception_ptr to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

Referenced by std::copy_exception().

2.9.3.4 template<typename _Ex > exception_ptr std::make_exception_ptr (_Ex *__ex*) [noexcept]

Obtain an exception_ptr pointing to a copy of the supplied object.

Definition at line 186 of file exception_ptr.h.

2.9.3.5 void std::rethrow_exception (exception_ptr)

Throw the object pointed to by the exception_ptr.

2.9.3.6 template<typename _Ex > void std::rethrow_if_nested (const _Ex & *__ex*) `[inline]`

If `__ex` is derived from nested_exception, `__ex.rethrow_nested()`.

Definition at line 146 of file nested_exception.h.

2.9.3.7 void std::rethrow_if_nested (const nested_exception & *__ex*) `[inline]`

Overload, See N2619.

Definition at line 154 of file nested_exception.h.

2.9.3.8 terminate_handler std::set_terminate (terminate_handler) `[noexcept]`

Takes a new handler function as an argument, returns the old function.

2.9.3.9 unexpected_handler std::set_unexpected (unexpected_handler) `[noexcept]`

Takes a new handler function as an argument, returns the old function.

2.9.3.10 void std::terminate ( ) `[noexcept]`

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

2.9.3.11 template<typename _Ex > void std::throw_with_nested (_Ex *__ex*) `[inline]`

If `__ex` is derived from nested_exception, `__ex`. Else, an implementation-defined object derived from both.

Definition at line 136 of file nested_exception.h.

2.9.3.12 bool std::uncaught_exception ( ) `[noexcept]`

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).'

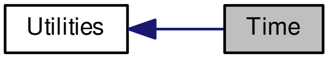
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

2.9.3.13 void std::unexpected ( )

The runtime will call this function if an exception is thrown which violates the function's exception specification.

2.10 Time

Collaboration diagram for Time:



Namespaces

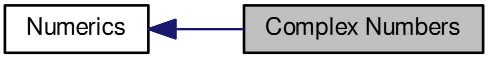
- namespace std::chrono

2.10.1 Detailed Description

Classes and functions for time.

2.11 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- struct std::complex< double >

  26.2.3 complex specializations complex<double> specialization
- struct std::complex< float >

  26.2.3 complex specializations complex<float> specialization
- struct std::complex< long double >

  26.2.3 complex specializations complex<long double> specialization

Functions

- constexpr **std::complex< float >::complex** (const complex< long double > &)
- template<typename _Tp >
  _Tp **std::__complex_abs** (const complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  _Tp **std::__complex_arg** (const complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::__complex_asinh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)

- template<typename _Tp >
  complex< _Tp > **std::__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_tanh** (const complex< _Tp > &__z)
- template<typename _Tp >
  _Tp std::abs (const complex< _Tp > &)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  _Tp std::arg (const complex< _Tp > &)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > std::conj (const complex< _Tp > &)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::conj** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< typename
  __gnu_cxx::__promote< _Tp >
  ::__type > **std::tr1::conj** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > std::cos (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::cosh (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::exp (const complex< _Tp > &)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp >
  constexpr _Tp **std::imag** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > std::log (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::log10 (const complex< _Tp > &)
- template<typename _Tp >
  _Tp std::norm (const complex< _Tp > &)
- complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &)
- template<typename _Up >
  complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &)
- template<typename _Tp >
  complex< _Tp > std::operator+ (const complex< _Tp > &__x)
- template<typename _Up >
  complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &)
- template<typename _Tp >
  complex< _Tp > std::operator- (const complex< _Tp > &__x)
- template<typename _Up >
  complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &)
- complex< _Tp > & std::complex< _Tp >::operator/= (const _Tp &)

- template<typename _Up >
  complex< _Tp > & std::complex< _Tp >::operator/= (const complex< _Up > &)
- template<typename _Tp , typename _CharT , class _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- complex< _Tp > & std::complex< _Tp >::operator= (const _Tp &)
- template<typename _Up >
  complex< _Tp > & std::complex< _Tp >::operator= (const complex< _Up > &)
- template<typename _Tp , typename _CharT , class _Traits >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template<typename _Tp >
  complex< _Tp > std::polar (const _Tp &, const _Tp &=0)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::tr1::polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp >
  complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp >
  complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::tr1::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp >
  constexpr _Tp **std::real** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > std::sin (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::sinh (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::sqrt (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::tan (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::tanh (const complex< _Tp > &)

- template<typename _Tp >
  complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)

2.11.1 Detailed Description

Classes and functions for complex numbers.

2.11.2 Function Documentation

2.11.2.1 template<typename _Tp > _Tp std::abs (const complex< _Tp > & __z) `[inline]`

Return magnitude of *z*.

Definition at line 599 of file complex.

Referenced by std::tr1::fabs(), std::fabs(), std::binomial_distribution< _IntType >::operator()(), and std::poisson_-distribution< _IntType >::operator()().

2.11.2.2 template<typename _Tp > std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > & __z) [inline]

acosh(__z) [8.1.5].

Definition at line 215 of file tr1/complex.

2.11.2.3 template<typename _Tp > _Tp std::arg (const complex< _Tp > & __z) [inline]

Return phase angle of *z*.

Definition at line 626 of file complex.

Referenced by std::arg().

2.11.2.4 template<typename _Tp > std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > & __z) [inline]

asinh(__z) [8.1.6].

Definition at line 254 of file tr1/complex.

2.11.2.5 template<typename _Tp > std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > & __z) [inline]

atanh(__z) [8.1.7].

Definition at line 298 of file tr1/complex.

2.11.2.6 template<typename _Tp > complex< _Tp > std::conj (const complex< _Tp > & __z) [inline]

Return complex conjugate of *z*.

Definition at line 672 of file complex.

2.11.2.7 template<typename _Tp > complex< _Tp > std::cos (const complex< _Tp > & __z) [inline]

Return complex cosine of *z*.

Definition at line 704 of file complex.

Referenced by std::polar().

2.11.2.8 template<typename _Tp > complex< _Tp > std::cosh (const complex< _Tp > & __z) [inline]

Return complex hyperbolic cosine of *z*.

Definition at line 734 of file complex.

2.11.2.9 template<typename _Tp > complex< _Tp > std::exp (const complex< _Tp > & __z) [inline]

Return complex base e exponential of *z*.

Definition at line 760 of file complex.

Referenced by std::pow().

2.11.2.10 template<typename _Tp > std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > & __z) [inline]

fabs(__z) [8.1.8].

Definition at line 307 of file tr1/complex.

References std::abs().

2.11.2.11 template< typename _Tp > complex< _Tp > std::log (const complex< _Tp > & *__z*) `[inline]`

Return complex natural logarithm of *z*.

Definition at line 787 of file complex.

Referenced by std::generate_canonical(), std::log10(), std::normal_distribution< _RealType >::operator()(), std::gamma_distribution< _RealType >::operator()(), std::binomial_distribution< _IntType >::operator()(), std::poisson_distribution< _IntType >::operator()(), and std::pow().

2.11.2.12 template< typename _Tp > complex< _Tp > std::log10 (const complex< _Tp > & *__z*) `[inline]`

Return complex base 10 logarithm of *z*.

Definition at line 792 of file complex.

References std::log().

2.11.2.13 template< typename _Tp > _Tp std::norm (const complex< _Tp > & *__z*) `[inline]`

Return *z* magnitude squared.

Definition at line 659 of file complex.

Referenced by std::complex< _Tp >::operator/=().

2.11.2.14 template< typename _Tp > constexpr bool std::operator!= (const complex< _Tp > & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return false if *x* is equal to *y*.

Definition at line 474 of file complex.

2.11.2.15 template< typename _Tp > constexpr bool std::operator!= (const complex< _Tp > & *__x*, const _Tp & *__y*) `[inline]`

Return false if *x* is equal to *y*.

Definition at line 479 of file complex.

2.11.2.16 template< typename _Tp > constexpr bool std::operator!= (const _Tp & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return false if *x* is equal to *y*.

Definition at line 484 of file complex.

2.11.2.17 template< typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return new complex value *x* times *y*.

Definition at line 384 of file complex.

2.11.2.18 template< typename _Tp > complex<_Tp> std::operator* (const complex< _Tp > & *__x*, const _Tp & *__y*) `[inline]`

Return new complex value *x* times *y*.

Definition at line 393 of file complex.

2.11.2.19 template<typename _Tp > complex<_Tp> std::operator* (const _Tp & __x, const complex< _Tp > & __y) `[inline]`

Return new complex value *x* times *y*.

Definition at line 402 of file complex.

2.11.2.20 template<typename _Tp > complex< _Tp > & complex< _Tp >::operator*= (const _Tp & __t)

Multiply this complex number by *t*.

Definition at line 243 of file complex.

2.11.2.21 template<typename _Tp > template<typename _Up > complex< _Tp > & complex< _Tp >::operator*= (const complex< _Up > & __z)

Multiply this complex number by *z*.

Definition at line 297 of file complex.

2.11.2.22 template<typename _Tp > complex<_Tp> std::operator+ (const complex< _Tp > & __x, const complex< _Tp > & __y) `[inline]`

Return new complex value *x* plus *y*.

Definition at line 324 of file complex.

2.11.2.23 template<typename _Tp > complex<_Tp> std::operator+ (const complex< _Tp > & __x, const _Tp & __y) `[inline]`

Return new complex value *x* plus *y*.

Definition at line 333 of file complex.

2.11.2.24 template<typename _Tp > complex<_Tp> std::operator+ (const _Tp & __x, const complex< _Tp > & __y) `[inline]`

Return new complex value *x* plus *y*.

Definition at line 342 of file complex.

2.11.2.25 template<typename _Tp > complex<_Tp> std::operator+ (const complex< _Tp > & __x) `[inline]`

Return *x*.

Definition at line 443 of file complex.

2.11.2.26 template<typename _Tp > template<typename _Up > complex< _Tp > & complex< _Tp >::operator+= (const complex< _Up > & __z)

Add *z* to this complex number.

Definition at line 274 of file complex.

2.11.2.27 template<typename _Tp > complex<_Tp> std::operator- (const complex< _Tp > & __x, const complex< _Tp > & __y) `[inline]`

Return new complex value *x* minus *y*.

Definition at line 354 of file complex.

2.11.2.28 template<typename _Tp > complex<_Tp> std::operator- (const complex< _Tp > & __x, const _Tp & __y) [inline]

Return new complex value *x* minus *y*.

Definition at line 363 of file complex.

2.11.2.29 template<typename _Tp > complex<_Tp> std::operator- (const _Tp & __x, const complex< _Tp > & __y) [inline]

Return new complex value *x* minus *y*.

Definition at line 372 of file complex.

2.11.2.30 template<typename _Tp > complex<_Tp> std::operator- (const complex< _Tp > & __x) [inline]

Return complex negation of *x*.

Definition at line 449 of file complex.

2.11.2.31 template<typename _Tp > template<typename _Up > complex< _Tp > & complex< _Tp >::operator-= (const complex< _Up > & __z)

Subtract *z* from this complex number.

Definition at line 285 of file complex.

2.11.2.32 template<typename _Tp > complex<_Tp> std::operator/ (const complex< _Tp > & __x, const complex< _Tp > & __y) [inline]

Return new complex value *x* divided by *y*.

Definition at line 414 of file complex.

2.11.2.33 template<typename _Tp > complex<_Tp> std::operator/ (const complex< _Tp > & __x, const _Tp & __y) [inline]

Return new complex value *x* divided by *y*.

Definition at line 423 of file complex.

2.11.2.34 template<typename _Tp > complex<_Tp> std::operator/ (const _Tp & __x, const complex< _Tp > & __y) [inline]

Return new complex value *x* divided by *y*.

Definition at line 432 of file complex.

2.11.2.35 template<typename _Tp > complex< _Tp > & complex< _Tp >::operator/= (const _Tp & __t)

Divide this complex number by *t*.

Definition at line 253 of file complex.

2.11.2.36 template<typename _Tp > template<typename _Up > complex< _Tp > & complex< _Tp >::operator/= (const complex< _Up > & __z)

Divide this complex number by *z*.

Definition at line 310 of file complex.

References std::norm().

2.11.2.37 template< typename _Tp , typename _CharT , class _Traits > basic_ostream< _CharT, _Traits >& std::operator<< (basic_ostream< _CharT, _Traits > & *__os*, const complex< _Tp > & *__x*)

Insertion operator for complex values.

Definition at line 524 of file complex.

References std::ios_base::flags(), std::basic_ios< _CharT, _Traits >::imbue(), std::ios_base::precision(), and std::basic_ostringstream< _CharT, _Traits, _Alloc >::str().

2.11.2.38 template< typename _Tp > complex< _Tp > & complex< _Tp >::operator= (const _Tp & *__t*)

Assign this complex number to scalar *t*.

Definition at line 233 of file complex.

2.11.2.39 template< typename _Tp > template< typename _Up > complex< _Tp > & complex< _Tp >::operator= (const complex< _Up > & *__z*)

Assign this complex number to complex *z*.

Definition at line 263 of file complex.

2.11.2.40 template< typename _Tp > constexpr bool std::operator== (const complex< _Tp > & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return true if *x* is equal to *y*.

Definition at line 456 of file complex.

2.11.2.41 template< typename _Tp > constexpr bool std::operator== (const complex< _Tp > & *__x*, const _Tp & *__y*) `[inline]`

Return true if *x* is equal to *y*.

Definition at line 461 of file complex.

2.11.2.42 template< typename _Tp > constexpr bool std::operator== (const _Tp & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return true if *x* is equal to *y*.

Definition at line 466 of file complex.

2.11.2.43 template< typename _Tp , typename _CharT , class _Traits > basic_istream< _CharT, _Traits >& std::operator>> (basic_istream< _CharT, _Traits > & *__is*, complex< _Tp > & *__x*)

Extraction operator for complex values.

Definition at line 491 of file complex.

References std::ios_base::failbit, std::basic_istream< _CharT, _Traits >::putback(), and std::basic_ios< _CharT, _Traits >::setstate().

2.11.2.44 template< typename _Tp > complex< _Tp > std::polar (const _Tp & *__rho*, const _Tp & *__theta* = 0) `[inline]`

Return complex with magnitude *rho* and angle *theta*.

Definition at line 667 of file complex.

References std::cos(), and std::sin().

Referenced by std::pow().

2.11.2.45 template< typename _Tp > complex< _Tp > std::pow (const complex< _Tp > & *__x*, const _Tp & *__y*)

Return *x* to the *y'th* power.

Definition at line 989 of file complex.

References std::exp(), std::log(), and std::polar().

Referenced by std::gamma_distribution< _RealType >::operator()(), std::pow(), and std::tr1::pow().

2.11.2.46 template< typename _Tp > complex< _Tp > std::pow (const complex< _Tp > & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return *x* to the *y'th* power.

Definition at line 1028 of file complex.

2.11.2.47 template< typename _Tp > complex< _Tp > std::pow (const _Tp & *__x*, const complex< _Tp > & *__y*) `[inline]`

Return *x* to the *y'th* power.

Definition at line 1034 of file complex.

References std::log(), std::polar(), and std::pow().

2.11.2.48 template< typename _Tp , typename _Up > std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type> std::tr1::pow (const std::complex< _Tp > & *__x*, const _Up & *__y*) `[inline]`

Additional overloads [8.1.9].

Definition at line 348 of file tr1/complex.

References std::pow().

2.11.2.49 template< typename _Tp > complex< _Tp > std::sin (const complex< _Tp > & *__z*) `[inline]`

Return complex sine of *z*.

Definition at line 822 of file complex.

Referenced by std::polar().

2.11.2.50 template< typename _Tp > complex< _Tp > std::sinh (const complex< _Tp > & *__z*) `[inline]`

Return complex hyperbolic sine of *z*.

Definition at line 852 of file complex.

2.11.2.51 template< typename _Tp > complex< _Tp > std::sqrt (const complex< _Tp > & *__z*) `[inline]`

Return complex square root of *z*.

Definition at line 896 of file complex.

Referenced by std::normal_distribution< _RealType >::operator()(), and std::student_t_distribution< _RealType >::operator()().

2.11.2.52 template<typename _Tp > complex< _Tp > std::tan (const complex< _Tp > & __z) [inline]

Return complex tangent of *z*.

Definition at line 923 of file complex.

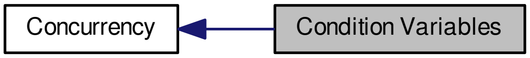
2.11.2.53 template<typename _Tp > complex< _Tp > std::tanh (const complex< _Tp > & __z) [inline]

Return complex hyperbolic tangent of *z*.

Definition at line 951 of file complex.

2.12 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class std::condition_variable

  condition_variable
- class std::condition_variable_any

  condition_variable_any

Enumerations

- enum std::cv_status

2.12.1 Detailed Description

Classes for condition_variable support.

2.12.2 Enumeration Type Documentation

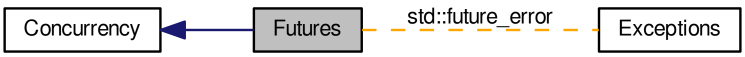
2.12.2.1 enum **std::cv_status**

cv_status

Definition at line 56 of file condition_variable.

2.13 Futures

Collaboration diagram for Futures:



Classes

- class std::future_error

  Exception type thrown by futures.
- struct std::is_error_code_enum< future_errc >

  Specialization.

Enumerations

- enum std::future_errc
- enum std::future_status
- enum std::launch

Functions

- template<typename _Fn , typename... _Args>
  future< typename result_of
  < _Fn(_Args...)>::type > **std::async** (launch __policy, _Fn &&__fn, _Args &&...__args)
- template<typename _Fn , typename... _Args>
  future< typename result_of
  < _Fn(_Args...)>::type > **std::async** (_Fn &&__fn, _Args &&...__args)
- const error_category & std::future_category () noexcept
- error_code std::make_error_code (future_errc __errc) noexcept
- error_condition std::make_error_condition (future_errc __errc) noexcept
- constexpr launch **std::operator&** (launch __x, launch __y)
- launch & **std::operator&=** (launch &__x, launch __y)
- constexpr launch **std::operator^** (launch __x, launch __y)
- launch & **std::operator^=** (launch &__x, launch __y)
- constexpr launch **std::operator|** (launch __x, launch __y)
- launch & **std::operator|=** (launch &__x, launch __y)
- constexpr launch **std::operator~** (launch __x)

2.13.1 Detailed Description

Classes for futures support.

2.13.2 Enumeration Type Documentation

2.13.2.1 enum std::future_errc

Error code for futures.

Definition at line 63 of file future.

2.13.2.2 enum std::future_status

Status code for futures.

Definition at line 162 of file future.

2.13.2.3 enum std::launch

Launch code for futures.

Definition at line 125 of file future.

2.13.3 Function Documentation

2.13.3.1 const error_category& std::future_category ( ) `[noexcept]`

Points to a statically-allocated object derived from error_category.

Referenced by std::make_error_code(), and std::make_error_condition().

2.13.3.2 error_code std::make_error_code (future_errc *__errc*) `[inline]`, `[noexcept]`

Overload for make_error_code.

Definition at line 81 of file future.

References std::future_category().

2.13.3.3 error_condition std::make_error_condition (future_errc *__errc*) `[inline]`, `[noexcept]`

Overload for make_error_condition.

Definition at line 86 of file future.

References std::future_category().

2.14 I/O

Classes

- class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

  Provides a layer of compatibility for C/POSIX.
  This GNU extension provides extensions for working with standard C FILE∗'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.
- class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

  Provides a layer of compatibility for C.
  This GNU extension provides extensions for working with standard C FILE∗'s. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.
- class std::basic_filebuf< _CharT, _Traits >

  The actual work of input and output (for files).
- class std::basic_fstream< _CharT, _Traits >

  Controlling input and output for files.
- class std::basic_ifstream< _CharT, _Traits >

  Controlling input for files.
- class std::basic_ofstream< _CharT, _Traits >

  Controlling output for files.
- class std::ios_base

  The base of the I/O class hierarchy.
  This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` *when they need to specify the full name of the various I/O flags (e.g., the openmodes).*

Typedefs

- typedef basic_filebuf< char > std::filebuf
- typedef basic_fstream< char > std::fstream
- typedef basic_ifstream< char > std::ifstream
- typedef basic_ios< char > std::ios
- typedef basic_iostream< char > std::iostream
- typedef basic_istream< char > std::istream
- typedef basic_istringstream< char > std::istringstream
- typedef basic_ofstream< char > std::ofstream
- typedef basic_ostream< char > std::ostream
- typedef basic_ostringstream< char > std::ostringstream
- typedef basic_streambuf< char > std::streambuf
- typedef basic_stringbuf< char > std::stringbuf
- typedef basic_stringstream< char > std::stringstream
- typedef basic_filebuf< wchar_t > std::wfilebuf
- typedef basic_fstream< wchar_t > std::wfstream
- typedef basic_ifstream< wchar_t > std::wifstream
- typedef basic_ios< wchar_t > std::wios
- typedef basic_iostream< wchar_t > std::wiostream
- typedef basic_istream< wchar_t > std::wistream
- typedef basic_istringstream
  < wchar_t > std::wistringstream
- typedef basic_ofstream< wchar_t > std::wofstream
- typedef basic_ostream< wchar_t > std::wostream

- typedef basic_ostringstream
  < wchar_t > std::wostringstream
- typedef basic_streambuf< wchar_t > std::wstreambuf
- typedef basic_stringbuf< wchar_t > std::wstringbuf
- typedef basic_stringstream
  < wchar_t > std::wstringstream

2.14.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is ios_base at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the basic_* I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char>  ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the <iosfwd> header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html`

2.14.2 Typedef Documentation

2.14.2.1 typedef basic_filebuf<char> std::filebuf

Class for `char` file buffers.

Definition at line 154 of file iosfwd.

2.14.2.2 typedef basic_fstream<char> std::fstream

Class for `char` mixed input and output file streams.

Definition at line 163 of file iosfwd.

2.14.2.3 typedef basic_ifstream<char> std::ifstream

Class for `char` input file streams.

Definition at line 157 of file iosfwd.

2.14.2.4 typedef basic_ios<char> std::ios

Base class for `char` streams.

Definition at line 123 of file iosfwd.

2.14.2.5 typedef basic_iostream<char> std::iostream

Base class for `char` mixed input and output streams.

Definition at line 139 of file iosfwd.

2.14.2.6 typedef basic_istream<char> std::istream

Base class for `char` input streams.

Definition at line 133 of file iosfwd.

2.14.2.7 typedef basic_istringstream<char> std::istringstream

Class for `char` input memory streams.

Definition at line 145 of file iosfwd.

2.14.2.8 typedef basic_ofstream<char> std::ofstream

Class for `char` output file streams.

Definition at line 160 of file iosfwd.

2.14.2.9 typedef basic_ostream<char> std::ostream

Base class for `char` output streams.

Definition at line 136 of file iosfwd.

2.14.2.10 typedef basic_ostringstream<char> std::ostringstream

Class for `char` output memory streams.

Definition at line 148 of file iosfwd.

2.14.2.11 typedef basic_streambuf<char> std::streambuf

Base class for `char` buffers.

Definition at line 130 of file iosfwd.

2.14.2.12 typedef basic_stringbuf<char> std::stringbuf

Class for `char` memory buffers.

Definition at line 142 of file iosfwd.

2.14.2.13 typedef basic_stringstream<char> std::stringstream

Class for `char` mixed input and output memory streams.

Definition at line 151 of file iosfwd.

2.14.2.14 typedef basic_filebuf<wchar_t> std::wfilebuf

Class for `wchar_t` file buffers.

Definition at line 194 of file iosfwd.

2.14.2.15 typedef basic_fstream<wchar_t> std::wfstream

Class for `wchar_t` mixed input and output file streams.

Definition at line 203 of file iosfwd.

2.14.2.16 typedef basic_ifstream<wchar_t> std::wifstream

Class for `wchar_t` input file streams.

Definition at line 197 of file iosfwd.

2.14.2.17 typedef basic_ios<wchar_t> std::wios

Base class for `wchar_t` streams.

Definition at line 167 of file iosfwd.

2.14.2.18 typedef basic_iostream<wchar_t> std::wiostream

Base class for `wchar_t` mixed input and output streams.

Definition at line 179 of file iosfwd.

2.14.2.19 typedef basic_istream<wchar_t> std::wistream

Base class for `wchar_t` input streams.

Definition at line 173 of file iosfwd.

2.14.2.20 typedef basic_istringstream<wchar_t> std::wistringstream

Class for `wchar_t` input memory streams.

Definition at line 185 of file iosfwd.

2.14.2.21 typedef basic_ofstream<wchar_t> std::wofstream

Class for `wchar_t` output file streams.

Definition at line 200 of file iosfwd.

2.14.2.22 typedef basic_ostream<wchar_t> std::wostream

Base class for `wchar_t` output streams.

Definition at line 176 of file iosfwd.

2.14.2.23 typedef basic_ostringstream<wchar_t> std::wostringstream

Class for `wchar_t` output memory streams.

Definition at line 188 of file iosfwd.

2.14.2.24 typedef basic_streambuf<wchar_t> std::wstreambuf

Base class for `wchar_t` buffers.

Definition at line 170 of file iosfwd.

2.14.2.25 typedef basic_stringbuf<wchar_t> std::wstringbuf

Class for `wchar_t` memory buffers.

Definition at line 182 of file iosfwd.

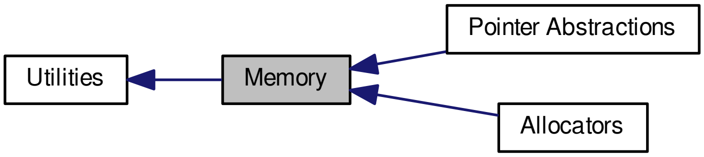
2.14.2.26 typedef basic_stringstream<wchar_t> std::wstringstream

Class for `wchar_t` mixed input and output memory streams.

Definition at line 191 of file iosfwd.

2.15 Memory

Collaboration diagram for Memory:



Modules

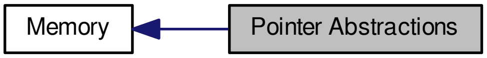
- Allocators
- Pointer Abstractions

2.15.1 Detailed Description

Components for memory allocation, deallocation, and management.

2.16 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- struct owner_less< _Tp >

  Primary template owner_less.
- struct std::default_delete< _Tp >

  Primary template, default_delete.
- struct std::default_delete< _Tp[]>

  Specialization, default_delete.
- class std::enable_shared_from_this< _Tp >

  Base class allowing use of member function shared_from_this.
- struct std::hash< shared_ptr< _Tp > >

  std::hash specialization for shared_ptr.
- struct std::hash< unique_ptr< _Tp, _Dp > >

  std::hash specialization for unique_ptr.
- struct std::owner_less< shared_ptr< _Tp > >

  Partial specialization of owner_less for shared_ptr.
- struct std::owner_less< weak_ptr< _Tp > >

  Partial specialization of owner_less for weak_ptr.
- struct std::pointer_traits< _Ptr >

  Uniform interface to all pointer-like types.
- struct std::pointer_traits< _Tp ∗ >

  Partial specialization for built-in pointers.
- class std::shared_ptr< _Tp >

  A smart pointer with reference-counted copy semantics.
- class std::unique_ptr< _Tp, _Dp >

  20.7.1.2 unique_ptr for single objects.
- class std::unique_ptr< _Tp[], _Dp >

  20.7.1.3 unique_ptr for array objects with a runtime length
- class std::weak_ptr< _Tp >

  A smart pointer with weak semantics.

Functions

- template<typename _Tp , typename _Alloc , typename... _Args>
  shared_ptr< _Tp > std::allocate_shared (const _Alloc &__a, _Args &&...__args)
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **std::const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **std::dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Del , typename _Tp , _Lock_policy _Lp>
  _Del ∗ std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Tp , typename... _Args>
  shared_ptr< _Tp > std::make_shared (_Args &&...__args)
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator!=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator!=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp , typename _Dp >
  bool **std::operator!=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator**< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator**< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator**< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator**< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator**< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator**< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator<=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator<=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator<=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept

- template<typename _Tp >
  bool **std::operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator==** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator==** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp , typename _Dp >
  bool **std::operator==** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator>** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator>=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator>=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator>=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **std::static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp >
  void **std::swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp , typename _Dp >
  void **std::swap** (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- template<typename _Tp >
  void **std::swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept

2.16.1 Detailed Description

Smart pointers, etc.

2.16.2 Function Documentation

2.16.2.1 template<typename _Tp , typename _Alloc , typename... _Args> shared_ptr<_Tp> std::allocate_shared (const _Alloc & *__a,* _Args &&... *__args*) [inline]

Create an object that is owned by a shared_ptr.

Parameters

__a	An allocator.
__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

An	exception thrown from *_Alloc::allocate* or from the constructor of *_Tp*.

A copy of *__a* will be used to allocate memory for the shared_ptr and the new object.

Definition at line 595 of file shared_ptr.h.

2.16.2.2 template<typename _Del , typename _Tp , _Lock_policy _Lp> _Del∗ std::get_deleter (const __shared_ptr< _Tp, _Lp > & *__p*) `[inline]`, `[noexcept]`

2.2.3.10 shared_ptr get_deleter (experimental)

Definition at line 76 of file shared_ptr.h.

2.16.2.3 template<typename _Tp , typename... _Args> shared_ptr<_Tp> std::make_shared (_Args &&... *__args*) `[inline]`

Create an object that is owned by a shared_ptr.

Parameters

__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

std::bad_alloc,or	an exception thrown from the constructor of *_Tp*.

Definition at line 610 of file shared_ptr.h.

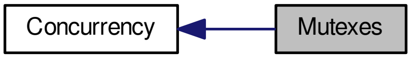
2.16.2.4 template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp> std::basic_ostream<_Ch, _Tr>& std::operator<< (std::basic_ostream< _Ch, _Tr > & *__os,* const __shared_ptr< _Tp, _Lp > & *__p*) `[inline]`

2.2.3.7 shared_ptr I/O

Definition at line 66 of file shared_ptr.h.

2.17 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct std::adopt_lock_t

  Assume the calling thread has already obtained mutex ownership and manage it.
- struct std::defer_lock_t

  Do not acquire ownership of the mutex.
- class std::lock_guard< _Mutex >

  Scoped lock idiom.
- class std::mutex

  mutex
- struct std::once_flag

  once_flag
- class std::recursive_mutex

  recursive_mutex
- struct std::try_to_lock_t

  Try to acquire ownership of the mutex without blocking.
- class std::unique_lock< _Mutex >

  unique_lock

Functions

- mutex & **std::__get_once_mutex** ()
- void **std::__once_proxy** (void)
- void **std::__set_once_functor_lock_ptr** (unique_lock< mutex > ∗)
- template<typename _Lock >
  unique_lock< _Lock > **std::__try_to_lock** (_Lock &__l)
- template<typename _Callable , typename... _Args>
  void std::call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)
- template<typename _L1 , typename _L2 , typename... _L3>
  void std::lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)
- template<typename _Mutex >
  void std::swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept
- template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
  int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)

Variables

- function< void()> **std::__once_functor**
- constexpr adopt_lock_t **std::adopt_lock**
- constexpr defer_lock_t **std::defer_lock**
- constexpr try_to_lock_t **std::try_to_lock**

2.17.1 Detailed Description

Classes for mutex support.

2.17.2 Function Documentation

2.17.2.1 template<typename _Callable , typename... _Args> void std::call_once (once_flag & *__once,* _Callable && *__f,* _Args &&... *__args*)

call_once

Definition at line 768 of file mutex.

2.17.2.2 template<typename _L1 , typename _L2 , typename... _L3> void std::lock (_L1 & *__l1,* _L2 & *__l2,* _L3 &... *__l3*)

Generic lock.

Parameters

__l1	Meets Mutex requirements (try_lock() may throw).
__l2	Meets Mutex requirements (try_lock() may throw).
__l3	Meets Mutex requirements (try_lock() may throw).

Exceptions

An	exception thrown by an argument's lock() or try_lock() member.

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to lock(), try_lock() and unlock(). If the call exits via an exception any locks that were obtained will be released.

Definition at line 706 of file mutex.

References std::tie().

2.17.2.3 template<typename _Mutex > void std::swap (unique_lock< _Mutex > & *__x,* unique_lock< _Mutex > & *__y*) `[inline]`, `[noexcept]`

Partial specialization for unique_lock objects.

Definition at line 605 of file mutex.

2.17.2.4 template<typename _Lock1 , typename _Lock2 , typename... _Lock3> int std::try_lock (_Lock1 & *__l1,* _Lock2 & *__l2,* _Lock3 &... *__l3*)

Generic try_lock.

Parameters

__l1	Meets Mutex requirements (try_lock() may throw).
__l2	Meets Mutex requirements (try_lock() may throw).
__l3	Meets Mutex requirements (try_lock() may throw).

Returns

> Returns -1 if all try_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

> Either all arguments are locked, or none will be.

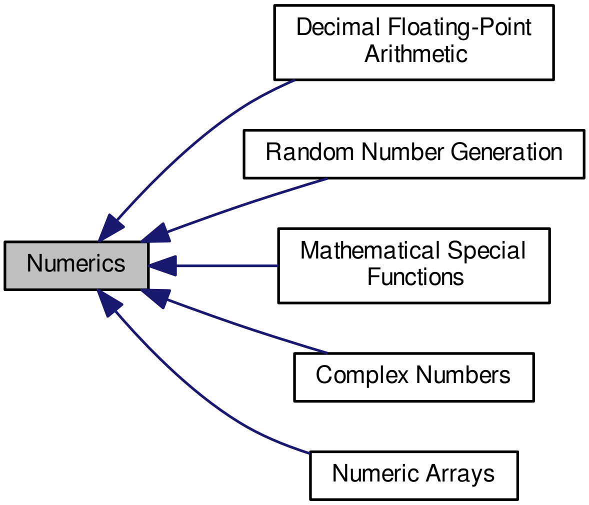
Sequentially calls try_lock() on each argument.

Definition at line 682 of file mutex.

References std::tie().

2.18 Numerics

Collaboration diagram for Numerics:



Modules

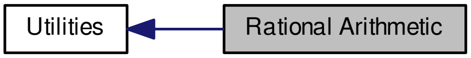
- Complex Numbers
- Decimal Floating-Point Arithmetic
- Mathematical Special Functions
- Numeric Arrays
- Random Number Generation

2.18.1 Detailed Description

Components for performing numeric operations. Includes support for for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and special math functions.

2.19 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Classes

- struct std::ratio< _Num, _Den >

  Provides compile-time rational arithmetic.
- struct std::ratio_equal< _R1, _R2 >

  ratio_equal
- struct std::ratio_not_equal< _R1, _R2 >

  ratio_not_equal

Typedefs

- template<typename _R1 , typename _R2 >
  using std::ratio_divide = typename __ratio_divide< _R1, _R2 >::type
- template<typename _R1 , typename _R2 >
  using std::ratio_multiply = typename __ratio_multiply< _R1, _R2 >::type
- typedef ratio< num, den > **std::ratio< _Num, _Den >::type**
- typedef ratio< __safe_multiply
  <(_R1::num/__gcd1),(_R2::num/__gcd2)>
  ::value, __safe_multiply
  <(_R1::den/__gcd2),(_R2::den/__gcd1)>
  ::value > **std::__ratio_multiply< _R1, _R2 >::type**
- typedef __ratio_multiply< _R1,
  ratio< _R2::den, _R2::num >
  >::type **std::__ratio_divide< _R1, _R2 >::type**

Variables

- static constexpr uintmax_t **std::__big_add< __hi1, __lo1, __hi2, __lo2 >::__hi**
- static constexpr uintmax_t **std::__big_sub< __hi1, __lo1, __hi2, __lo2 >::__hi**
- static constexpr uintmax_t **std::__big_mul< __x, __y >::__hi**
- static constexpr uintmax_t **std::__big_add< __hi1, __lo1, __hi2, __lo2 >::__lo**
- static constexpr uintmax_t **std::__big_sub< __hi1, __lo1, __hi2, __lo2 >::__lo**
- static constexpr uintmax_t **std::__big_mul< __x, __y >::__lo**
- static constexpr uintmax_t **std::__big_div_impl< __n1, __n0, __d >::__quot**
- static constexpr uintmax_t **std::__big_div< __n1, __n0, __d >::__quot_hi**
- static constexpr uintmax_t **std::__big_div< __n1, __n0, __d >::__quot_lo**

- static constexpr uintmax_t **std::__big_div_impl< __n1, __n0, __d >::__rem**
- static constexpr uintmax_t **std::__big_div< __n1, __n0, __d >::__rem**
- static constexpr intmax_t **std::ratio< _Num, _Den >::den**
- static constexpr intmax_t **std::__ratio_multiply< _R1, _R2 >::den**
- static constexpr intmax_t **std::__ratio_divide< _R1, _R2 >::den**
- static constexpr intmax_t **std::ratio< _Num, _Den >::num**
- static constexpr intmax_t **std::__ratio_multiply< _R1, _R2 >::num**
- static constexpr intmax_t **std::__ratio_divide< _R1, _R2 >::num**
- static const intmax_t **std::__safe_multiply< _Pn, _Qn >::value**

2.19.1 Detailed Description

Compile time representation of finite rational numbers.

2.19.2 Typedef Documentation

2.19.2.1 template<typename _R1 , typename _R2 > using std::ratio_divide = typedef typename __ratio_divide<_R1, _R2>::type

ratio_divide

Definition at line 336 of file ratio.

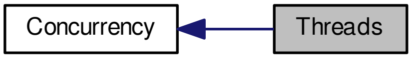
2.19.2.2 template<typename _R1 , typename _R2 > using std::ratio_multiply = typedef typename __ratio_multiply<_R1, _R2>::type

ratio_multiply

Definition at line 313 of file ratio.

2.20 Threads

Collaboration diagram for Threads:



Namespaces

- namespace std::this_thread

Classes

- struct std::hash< thread::id >

  std::hash specialization for thread::id.
- class std::thread

  thread
- class std::thread::id

  thread::id

Functions

- bool **std::operator!=** (thread::id __x, thread::id __y) noexcept
- template<class _CharT , class _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- bool **std::operator<=** (thread::id __x, thread::id __y) noexcept
- bool **std::operator>** (thread::id __x, thread::id __y) noexcept
- bool **std::operator>=** (thread::id __x, thread::id __y) noexcept
- void **std::swap** (thread &__x, thread &__y) noexcept

2.20.1 Detailed Description

Classes for thread support.

2.21 Metaprogramming

Collaboration diagram for Metaprogramming:

Classes

- struct __reflection_typelist< _Elements >
- struct common_type< _Tp >

  common_type
- class result_of< _Signature >

  result_of
- struct std::add_const< _Tp >

  add_const
- struct std::add_cv< _Tp >

  add_cv
- struct std::add_lvalue_reference< _Tp >

  add_lvalue_reference
- struct std::add_pointer< _Tp >

  add_pointer
- struct std::add_volatile< _Tp >

  add_volatile
- struct std::aligned_storage< _Len, _Align >

  Alignment type.
- struct std::alignment_of< _Tp >

  alignment_of
- class std::decay< _Tp >

  decay
- struct std::enable_if< bool, _Tp >

  Define a member typedef `type` *only if a boolean constant is true.*
- struct std::has_trivial_copy_assign< _Tp >

  has_trivial_copy_assign (temporary legacy)
- struct std::has_trivial_copy_constructor< _Tp >

  has_trivial_copy_constructor (temporary legacy)
- struct std::has_trivial_default_constructor< _Tp >

  has_trivial_default_constructor (temporary legacy)
- struct std::has_virtual_destructor< _Tp >

  has_virtual_destructor
- struct std::integral_constant< _Tp, __v >

  integral_constant
- struct std::is_abstract< _Tp >

is_abstract

- struct std::is_arithmetic< _Tp >

  is_arithmetic
- struct std::is_array< typename >

  is_array
- struct std::is_assignable< _Tp, _Up >

  is_assignable
- struct std::is_class< _Tp >

  is_class
- struct std::is_compound< _Tp >

  is_compound
- struct std::is_const< typename >

  is_const
- struct std::is_constructible< _Tp, _Args >

  is_constructible
- struct std::is_convertible< _From, _To >

  is_convertible
- struct std::is_copy_assignable< _Tp >

  is_copy_assignable
- struct std::is_copy_constructible< _Tp >

  is_copy_constructible
- struct std::is_default_constructible< _Tp >

  is_default_constructible
- struct std::is_destructible< _Tp >

  is_destructible
- struct std::is_empty< _Tp >

  is_empty
- struct std::is_enum< _Tp >

  is_enum
- struct std::is_floating_point< _Tp >

  is_floating_point
- struct std::is_fundamental< _Tp >

  is_fundamental
- struct std::is_integral< _Tp >

  is_integral
- struct std::is_literal_type< _Tp >

  is_literal_type
- struct std::is_lvalue_reference< typename >

  is_lvalue_reference
- struct std::is_member_function_pointer< _Tp >

  is_member_function_pointer
- struct std::is_member_object_pointer< _Tp >

  is_member_object_pointer
- struct std::is_move_assignable< _Tp >

  is_move_assignable
- struct std::is_move_constructible< _Tp >

  is_move_constructible

- struct std::is_nothrow_assignable< _Tp, _Up >

  is_nothrow_assignable
- struct std::is_nothrow_constructible< _Tp, _Args >

  is_nothrow_constructible
- struct std::is_nothrow_copy_assignable< _Tp >

  is_nothrow_copy_assignable
- struct std::is_nothrow_copy_constructible< _Tp >

  is_nothrow_copy_constructible
- struct std::is_nothrow_default_constructible< _Tp >

  is_nothrow_default_constructible
- struct std::is_nothrow_destructible< _Tp >

  is_nothrow_destructible
- struct std::is_nothrow_move_assignable< _Tp >

  is_nothrow_move_assignable
- struct std::is_nothrow_move_constructible< _Tp >

  is_nothrow_move_constructible
- struct std::is_object< _Tp >

  is_object
- struct std::is_pod< _Tp >

  is_pod
- struct std::is_pointer< _Tp >

  is_pointer
- struct std::is_polymorphic< _Tp >

  is_polymorphic
- struct std::is_reference< _Tp >

  is_reference
- struct std::is_rvalue_reference< typename >

  is_rvalue_reference
- struct std::is_scalar< _Tp >

  is_scalar
- struct std::is_signed< _Tp >

  is_signed
- struct std::is_standard_layout< _Tp >

  is_standard_layout
- struct std::is_trivial< _Tp >

  is_trivial
- struct std::is_trivially_destructible< _Tp >

  is_trivially_constructible (still unimplemented)
- struct std::is_union< _Tp >

  is_union
- struct std::is_unsigned< _Tp >

  is_unsigned
- struct std::is_void< _Tp >

  is_void
- struct std::is_volatile< typename >

  is_volatile
- struct std::make_signed< _Tp >

make_signed
- struct std::make_unsigned< _Tp >

  make_unsigned
- struct std::rank< typename >

  rank
- class std::reference_wrapper< _Tp >

  Primary class template for reference_wrapper.
- struct std::remove_const< _Tp >

  remove_const
- struct std::remove_extent< _Tp >

  remove_extent
- struct std::remove_pointer< _Tp >

  remove_pointer
- struct std::remove_volatile< _Tp >

  remove_volatile
- struct std::tr2::__reflection_typelist< _First, _Rest...>

  Partial specialization.
- struct std::tr2::__reflection_typelist<>

  Specialization for an empty typelist.
- struct std::tr2::bases< _Tp >

  Sequence abstraction metafunctions for manipulating a typelist.
- struct std::tr2::direct_bases< _Tp >

  Enumerate all the direct base classes of a class. Form of a typelist.
- struct std::underlying_type< _Tp >

  The underlying type of an enum.

Typedefs

- template<typename... _Cond>
  using **std::_Require** = typename enable_if< __and_< _Cond...>::value >::type
- typedef integral_constant
  < bool, false > std::false_type
- typedef integral_constant
  < bool, true > std::true_type

Variables

- static constexpr _Tp **std::integral_constant< _Tp, __v >::value**

2.21.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

2.21.2 Typedef Documentation

2.21.2.1 typedef integral_constant<bool, false> std::false_type

The type used as a compile-time boolean with false value.

Definition at line 72 of file type_traits.

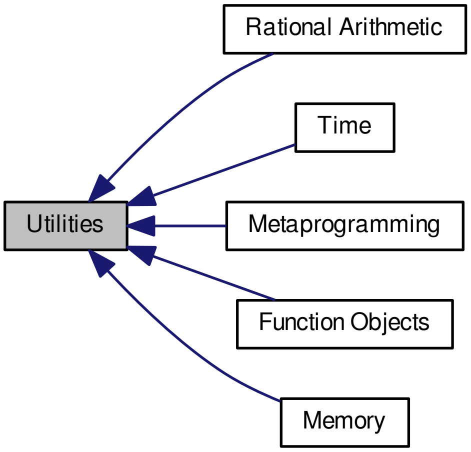
2.21.2.2 typedef integral_constant<bool, true> std::true_type

The type used as a compile-time boolean with true value.

Definition at line 69 of file type_traits.

2.22 Utilities

Collaboration diagram for Utilities:



Modules

- Function Objects
- Memory
- Metaprogramming
- Rational Arithmetic
- Time

Classes

- struct _Tuple_impl< _Idx, _Elements >
- struct std::_Tuple_impl< _Idx >
- struct std::_Tuple_impl< _Idx, _Head, _Tail...>
- struct std::pair< _T1, _T2 >

  Struct holding two objects of arbitrary type.
- struct std::piecewise_construct_t

  piecewise_construct_t
- class std::tuple< _Elements >

  Primary class template, tuple.
- class std::tuple< _T1, _T2 >

  Partial specialization, 2-element tuple. Includes construction and assignment from a pair.
- struct std::tuple_element< 0, tuple< _Head, _Tail...> >
- struct std::tuple_element< __i, tuple< _Head, _Tail...> >
- struct std::tuple_size< tuple< _Elements...> >

class tuple_size

- struct std::type_index

  Class type_index
  The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).
- struct std::uses_allocator< tuple< _Types...>, _Alloc >

  Partial specialization for tuples.
- struct tuple_element< __i, _Tp >

  Gives the type of the ith element of a given tuple type.
- struct tuple_size< _Tp >

  Finds the size of a given tuple type.

Typedefs

- template<typename _Tp >
  using **std::__empty_not_final** = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp >>::type

Functions

- template<typename... _Args1, typename... _Args2>
  std::pair< **_T1, _T2** >**::pair** (piecewise_construct_t, tuple< _Args1...>, tuple< _Args2...>)
- template<typename _Tp >
  _Tp ∗ std::__addressof (_Tp &__r) noexcept
- template<std::size_t __i, typename _Head , typename... _Tail>
  constexpr __add_ref< _Head >::type **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail...> &__t) noexcept
- template<std::size_t __i, typename _Head , typename... _Tail>
  constexpr __add_c_ref< _Head >
  ::type **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail...> &__t) noexcept
- template<typename _Tp >
  _Tp ∗ std::addressof (_Tp &__r) noexcept
- template<typename _Tp >
  add_rvalue_reference< _Tp >::type std::declval () noexcept
- template<typename _Tp >
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename _Tp >
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename... _Elements>
  tuple< _Elements &&...> **std::forward_as_tuple** (_Elements &&...__args) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_ref< typename
  tuple_element< __i, tuple
  < _Elements...> >::type >
  ::type **std::get** (tuple< _Elements...> &__t) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_c_ref
  < typename tuple_element< __i,
  tuple< _Elements...> >::type >
  ::type **std::get** (const tuple< _Elements...> &__t) noexcept

- template<std::size_t __i, typename... _Elements>
  constexpr __add_r_ref
  < typename tuple_element< __i,
  tuple< _Elements...> >::type >
  ::type **std::get** (tuple< _Elements...> &&__t) noexcept
- template<class _T1 , class _T2 >
  constexpr pair< typename
  __decay_and_strip< _T1 >
  ::__type, typename
  __decay_and_strip< _T2 >
  ::__type > std::make_pair (_T1 &&__x, _T2 &&__y)
- template<typename... _Elements>
  constexpr tuple< typename
  __decay_and_strip< _Elements >
  ::__type...> **std::make_tuple** (_Elements &&...__args)
- template<typename _Tp >
  constexpr
  std::remove_reference< _Tp >
  ::type && std::move (_Tp &&__t) noexcept
- template<typename _Tp >
  constexpr conditional
  < __move_if_noexcept_cond< _Tp >
  ::value, const _Tp &, _Tp && >
  ::type std::move_if_noexcept (_Tp &__x) noexcept
- template<class _T1 , class _T2 >
  constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator!=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<class _T1 , class _T2 >
  constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<class _T1 , class _T2 >
  constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**<**=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<class _T1 , class _T2 >
  constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator==** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<class _T1 , class _T2 >
  constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<class _T1 , class _T2 >
  constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**>**=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<class _T1 , class _T2 >
  void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp >
  void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >

- template<typename _Tp , size_t _Nm>
  void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(∗__a
- template<typename... _Elements>
  void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  tuple< _Elements &...> std::tie (_Elements &...__args) noexcept
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto std::tuple_cat (_Tpls &&...__tpls) -> typename __tuple_cat_result< _Tpls...>::__type

Variables

- **std::__a**
- void ∗ **std::__b**
- void
  is_nothrow_move_assignable
  < _Tp >::value _Tp **std::__tmp**
- const _Swallow_assign **std::ignore**
- constexpr piecewise_construct_t std::piecewise_construct

2.22.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

2.22.2 Function Documentation

2.22.2.1 template<typename _Tp > _Tp∗ std::__addressof (_Tp & __r) [inline], [noexcept]

Same as C++11 std::addressof.

Definition at line 47 of file move.h.

Referenced by std::addressof(), std::begin(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::data(), std::end(), std::forward_list< _Tp, _Alloc >::remove(), and std::list< _Tp, _Alloc >::remove().

2.22.2.2 template<typename _Tp > _Tp∗ std::addressof (_Tp & __r) [inline], [noexcept]

Returns the actual address of the object or function referenced by r, even in the presence of an overloaded operator&.

Parameters

__r	Reference to an object or function.

Returns

The actual address.

Definition at line 135 of file move.h.

References std::__addressof().

Referenced by std::pointer_traits< _Tp ∗ >::pointer_to().

2.22.2.3 template<typename _Tp > add_rvalue_reference< _Tp >::type std::declval ( ) [inline], [noexcept]

Utility to simplify expressions used in unevaluated operands.

Definition at line 1869 of file type_traits.

2.22.2.4 template<typename _Tp > constexpr _Tp&& std::forward (typename std::remove_reference< _Tp >::type & __t) [noexcept]

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 76 of file move.h.

2.22.2.5 template<typename _Tp > constexpr _Tp&& std::forward (typename std::remove_reference< _Tp >::type && __t) [noexcept]

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 87 of file move.h.

2.22.2.6 template<class _T1 , class _T2 > constexpr pair<typename __decay_and_strip<_T1>::__type, typename __decay_and_strip<_T2>::__type> std::make_pair (_T1 && __x, _T2 && __y)

A convenience wrapper for creating a pair from two objects.

Parameters

__x	The first object.
__y	The second object.

Returns

A newly-constructed pair<> object of the appropriate type.

The standard requires that the objects be passed by reference-to-const, but LWG issue #181 says they should be passed by const value. We follow the LWG by default.

Definition at line 276 of file stl_pair.h.

Referenced by __gnu_parallel::__find_template(), __gnu_debug::__get_distance(), __gnu_parallel::__parallel_merge_advance(), __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__qsb_local_sort_with_helping(), __gnu_parallel::__find_if_selector::_M_sequential_algorithm(), __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm(), __gnu_parallel::__find_first_of_selector< _FIterator >::_M_sequential_algorithm(), std::minmax_element(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::parallel_multiway_merge(), __gnu_parallel::parallel_sort_mwms_pu(), and __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix().

2.22.2.7 template<typename _Tp > constexpr std::remove_reference<_Tp>::type&& std::move (_Tp && __t) [noexcept]

Convert a value to an rvalue.

Parameters

__t	A thing of arbitrary type.

Returns

The parameter cast to an rvalue-reference to allow moving it.

Definition at line 101 of file move.h.

2.22.2.8 template<typename _Tp > constexpr conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp&, _Tp&&>::type std::move_if_noexcept (_Tp & __x) [inline], [noexcept]

Conditionally convert a value to an rvalue.

Parameters

__x	A thing of arbitrary type.

Returns

The parameter, possibly cast to an rvalue-reference.

Same as std::move unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Definition at line 121 of file move.h.

2.22.2.9 template<class _T1 , class _T2 > constexpr bool std::operator!= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]

Uses `operator==` to find the result.

Definition at line 227 of file stl_pair.h.

2.22.2.10 template<class _T1 , class _T2 > constexpr bool std::operator< (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]

`http://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html`

Definition at line 220 of file stl_pair.h.

References std::pair< _T1, _T2 >::first.

2.22.2.11 template<class _T1 , class _T2 > constexpr bool std::operator<= (const pair< _T1, _T2 > & __x, const pair< _T1, _T2 > & __y) [inline]

Uses `operator<` to find the result.

Definition at line 239 of file stl_pair.h.

2.22.2.12 template< class _T1 , class _T2 > constexpr bool std::operator== (const pair< _T1, _T2 > & *__x*, const pair< _T1, _T2 > & *__y*) [inline]

Two pairs of the same type are equal iff their members are equal.

Definition at line 214 of file stl_pair.h.

References std::pair< _T1, _T2 >::first, and std::pair< _T1, _T2 >::second.

2.22.2.13 template< class _T1 , class _T2 > constexpr bool std::operator> (const pair< _T1, _T2 > & *__x*, const pair< _T1, _T2 > & *__y*) [inline]

Uses `operator<` to find the result.

Definition at line 233 of file stl_pair.h.

2.22.2.14 template< class _T1 , class _T2 > constexpr bool std::operator>= (const pair< _T1, _T2 > & *__x*, const pair< _T1, _T2 > & *__y*) [inline]

Uses `operator<` to find the result.

Definition at line 245 of file stl_pair.h.

2.22.2.15 template< class _T1 , class _T2 > void std::swap (pair< _T1, _T2 > & *__x*, pair< _T1, _T2 > & *__y*) [inline], [noexcept]

See std::pair::swap().

Definition at line 254 of file stl_pair.h.

2.22.2.16 template< typename _Tp > void std::swap (_Tp & *__a*, _Tp & *__b*) const [inline], [noexcept]

Swaps two values.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.

Returns

Nothing.

2.22.2.17 template< typename _Tp , size_t _Nm> void std::swap (_Tp(&) *__a[_Nm]*, _Tp(&) *__b[_Nm]*) [inline], [noexcept]

Swap the contents of two arrays.

2.22.2.18 template< typename... _Elements> void std::swap (tuple< _Elements...> & *__x*, tuple< _Elements...> & *__y*) [inline], [noexcept]

swap

Definition at line 1048 of file tuple.

2.22.2.19 template< typename... _Elements> tuple< _Elements&...> std::tie (_Elements &... *__args*) [inline], [noexcept]

tie

Definition at line 1042 of file tuple.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::lock(), and std::try_lock().

2.22.2.20 template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type> constexpr auto std::tuple_cat (_Tpls &&... *__tpls*) -> typename __tuple_cat_result<_Tpls...>::__type

tuple_cat

Definition at line 1030 of file tuple.

2.22.3 Variable Documentation

2.22.3.1 constexpr piecewise_construct_t std::piecewise_construct

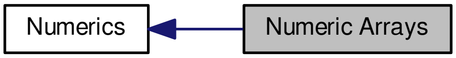
piecewise_construct

Definition at line 79 of file stl_pair.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::operator[ ]().

2.23 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class std::gslice

  Class defining multi-dimensional subset of an array.
- class std::gslice_array< _Tp >

  Reference to multi-dimensional subset of an array.
- class std::indirect_array< _Tp >

  Reference to arbitrary subset of an array.
- class std::mask_array< _Tp >

  Reference to selected subset of an array.
- class std::slice

  Class defining one-dimensional subset of an array.
- class std::slice_array< _Tp >

  Reference to one-dimensional subset of an array.
- class std::valarray< _Tp >

  Smart array designed to support numeric processing.

Macros

- #define **_DEFINE_BINARY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT**(_Op, _Name)
- #define **_DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT**(_Op, _Name)
- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_VALARRAY_UNARY_OPERATOR**(_Op, _Name)

Functions

- std::gslice::gslice ()
- std::gslice::gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- std::gslice::gslice (const gslice &)
- std::gslice_array< _Tp >::gslice_array (const gslice_array &)
- std::indirect_array< _Tp >::indirect_array (const indirect_array &)

- std::mask_array< _Tp >::mask_array (const mask_array &)
- std::slice::slice ()
- std::slice::slice (size_t __o, size_t __d, size_t __s)
- std::slice_array< _Tp >::slice_array (const slice_array &)
- std::valarray< _Tp >::valarray ()
- std::valarray< _Tp >::valarray (size_t)
- std::valarray< _Tp >::valarray (const _Tp &, size_t)
- std::valarray< _Tp >::valarray (const valarray &)
- std::valarray< _Tp >::valarray (valarray &&) noexcept
- std::valarray< _Tp >::valarray (const slice_array< _Tp > &)
- std::valarray< _Tp >::valarray (const gslice_array< _Tp > &)
- std::valarray< _Tp >::valarray (const mask_array< _Tp > &)
- std::valarray< _Tp >::valarray (const indirect_array< _Tp > &)
- std::valarray< _Tp >::valarray (initializer_list< _Tp >)
- template<class _Dom >
  std::valarray< _Tp >::valarray (const _Expr< _Dom, _Tp > &__e)
- template<typename _Tp>
  std::valarray< _Tp >::valarray (const _Tp ∗__restrict__ __p, size_t __n)
- std::gslice::∼gslice ()
- _Expr< _ValFunClos< _ValArray,
  _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp func(_Tp)) const
- _Expr< _RefFunClos< _ValArray,
  _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp func(const _Tp &)) const
- template<class _Tp >
  _Tp ∗ std::begin (valarray< _Tp > &__va)
- template<class _Tp >
  const _Tp ∗ std::begin (const valarray< _Tp > &__va)
- valarray< _Tp > std::valarray< _Tp >::cshift (int __n) const
- template<class _Tp >
  _Tp ∗ std::end (valarray< _Tp > &__va)
- template<class _Tp >
  const _Tp ∗ std::end (const valarray< _Tp > &__va)
- _Tp std::valarray< _Tp >::max () const
- _Tp std::valarray< _Tp >::min () const
- _UnaryOp< __logical_not >::_Rt std::valarray< _Tp >::operator! () const
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _Constant, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **std::operator!=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _Constant,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **std::operator!=** (const _Tp &__t, const valarray< _Tp > &__v)

- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **std::operator!=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **std::operator%** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **std::operator%** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **std::operator%** (const _Tp &__t, const valarray< _Tp > &__v)
- void std::gslice_array< _Tp >::operator%= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator%= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator%= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator%=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator%=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator%=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator%= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator%=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator%= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator%= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator%=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **std::operator&** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **std::operator&** (const _Tp &__t, const valarray< _Tp > &__v)

- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **std::operator&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **std::operator&&** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **std::operator&&** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **std::operator&&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- void std::gslice_array< _Tp >::operator&= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator&= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator&= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator&=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator&=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator&=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator&= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator&=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator&= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator&= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator&=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **std::operator*** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **std::operator*** (const _Tp &__t, const valarray< _Tp > &__v)

- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **std::operator∗** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- void std::gslice_array< _Tp >::operator∗= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator∗= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator∗= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator∗=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator∗=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator∗=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator∗= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator∗=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator∗= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator∗= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator∗=** (const _Expr< _Dom, _Tp > &)
- _UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+ () const
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **std::operator+** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **std::operator+** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **std::operator+** (const _Tp &__t, const valarray< _Tp > &__v)
- void std::gslice_array< _Tp >::operator+= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator+= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator+= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator+=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator+=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator+=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator+= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator+=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator+= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator+= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator+=** (const _Expr< _Dom, _Tp > &)

- _UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- () const
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **std::operator-** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **std::operator-** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **std::operator-** (const _Tp &__t, const valarray< _Tp > &__v)
- void std::gslice_array< _Tp >::operator-= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator-= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator-= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator-=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator-=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator-=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator-= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator-=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator-= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator-= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator-=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **std::operator/** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **std::operator/** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **std::operator/** (const _Tp &__t, const valarray< _Tp > &__v)
- void std::gslice_array< _Tp >::operator/= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator/= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator/= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator/=** (const _Expr< _Dom, _Tp > &) const

- template<class _Dom >
  void **std::mask_array< _Tp >::operator/=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array< _Tp >::operator/=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator/= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array< _Tp >::operator/=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator/= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator/= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray< _Tp >::operator/=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **std::operator<** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **std::operator<** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **std::operator<** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **std::operator<<** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **std::operator<<** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **std::operator<<** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- void std::gslice_array< _Tp >::operator<<= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator<<= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator<<= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array< _Tp >::operator<<=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array< _Tp >::operator<<=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array< _Tp >::operator<<=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator<<= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array< _Tp >::operator<<=** (const _Expr< _Dom, _Tp > &) const

- valarray< _Tp > & std::valarray< _Tp >::operator<<= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator<<= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator**<<**=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **std::operator**<**=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **std::operator**<**=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **std::operator**<**=** (const valarray< _Tp > &__v, const _Tp &__t)
- gslice_array & std::gslice_array< _Tp >::operator= (const gslice_array &)
- indirect_array & std::indirect_array< _Tp >::operator= (const indirect_array &)
- mask_array & std::mask_array< _Tp >::operator= (const mask_array &)
- void std::gslice_array< _Tp >::operator= (const valarray< _Tp > &) const
- void **std::mask_array**< **_Tp** >**::operator=** (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator= (const valarray< _Tp > &) const
- gslice & std::gslice::operator= (const gslice &)
- void std::gslice_array< _Tp >::operator= (const _Tp &) const
- void std::mask_array< _Tp >::operator= (const _Tp &) const
- void std::indirect_array< _Tp >::operator= (const _Tp &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator=** (const _Expr< _Dom, _Tp > &) const
- slice_array & std::slice_array< _Tp >::operator= (const slice_array &)
- void std::slice_array< _Tp >::operator= (const valarray< _Tp > &) const
- void std::slice_array< _Tp >::operator= (const _Tp &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator=** (const _Expr< _Dom, _Tp > &) const
- template<class _Ex >
  void **std::mask_array**< **_Tp** >**::operator=** (const _Expr< _Ex, _Tp > &__e) const
- valarray< _Tp > & std::valarray< _Tp >::operator= (const valarray< _Tp > &__v)
- valarray< _Tp > & std::valarray< _Tp >::operator= (valarray< _Tp > &&__v) noexcept
- valarray< _Tp > & std::valarray< _Tp >::operator= (const _Tp &__t)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const slice_array< _Tp > &__sa)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const gslice_array< _Tp > &__ga)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const mask_array< _Tp > &__ma)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const indirect_array< _Tp > &__ia)
- valarray & std::valarray< _Tp >::operator= (initializer_list< _Tp > __l)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator=** (const _Expr< _Dom, _Tp > &)

- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **std::operator==** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **std::operator==** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **std::operator==** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **std::operator**> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **std::operator**> (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **std::operator**> (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _Constant,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **std::operator**>**=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _Constant, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **std::operator**>**=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **std::operator**>**=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >

_Expr< _BinClos< __shift_right,
_ValArray, _Constant, _Tp, _Tp >
, typename __fun
< __shift_right, _Tp >
::result_type > **std::operator>>** (const valarray< _Tp > &__v, const _Tp &__t)

- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **std::operator>>** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **std::operator>>** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- void std::gslice_array< _Tp >::operator>>= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator>>= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator>>= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator>>=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator>>=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator>>=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator>>= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator>>=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator>>= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator>>=** (const _Expr< _Dom, _Tp > &)
- _Tp & std::valarray< _Tp >::operator[] (size_t __i)
- const _Tp & **std::valarray**< **_Tp** >**::operator[]** (size_t) const
- _Expr< _SClos< _ValArray, _Tp >
  , _Tp > std::valarray< _Tp >::operator[] (slice __s) const
- slice_array< _Tp > std::valarray< _Tp >::operator[] (slice __s)
- _Expr< _GClos< _ValArray, _Tp >
  , _Tp > std::valarray< _Tp >::operator[] (const gslice &__s) const
- gslice_array< _Tp > std::valarray< _Tp >::operator[] (const gslice &__s)
- valarray< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m) const
- mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m)
- _Expr< _IClos< _ValArray, _Tp >
  , _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i) const
- indirect_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **std::operator^** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **std::operator^** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **std::operator^** (const _Tp &__t, const valarray< _Tp > &__v)
- void std::gslice_array< _Tp >::operator^= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator^= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator^= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator^=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator^=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator^=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator^= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::slice_array**< **_Tp** >**::operator^=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator^= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray**< **_Tp** >**::operator^=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **std::operator|** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **std::operator|** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **std::operator|** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- void std::gslice_array< _Tp >::operator|= (const valarray< _Tp > &) const
- void std::mask_array< _Tp >::operator|= (const valarray< _Tp > &) const
- void std::indirect_array< _Tp >::operator|= (const valarray< _Tp > &) const
- template<class _Dom >
  void **std::gslice_array**< **_Tp** >**::operator|=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::indirect_array**< **_Tp** >**::operator|=** (const _Expr< _Dom, _Tp > &) const
- template<class _Dom >
  void **std::mask_array**< **_Tp** >**::operator|=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator|= (const valarray< _Tp > &) const

- template<class _Dom >
  void **std::slice_array< _Tp >::operator|=** (const _Expr< _Dom, _Tp > &) const
- valarray< _Tp > & std::valarray< _Tp >::operator|= (const _Tp &)
- valarray< _Tp > & std::valarray< _Tp >::operator|= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **std::valarray< _Tp >::operator|=** (const _Expr< _Dom, _Tp > &)
- template<typename _Tp >
  _Expr< _BinClos< __logical_or,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __logical_or,
  _Tp >::result_type > **std::operator||** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __logical_or,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __logical_or,
  _Tp >::result_type > **std::operator||** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __logical_or,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __logical_or,
  _Tp >::result_type > **std::operator||** (const _Tp &__t, const valarray< _Tp > &__v)
- _UnaryOp< __bitwise_not >::_Rt std::valarray< _Tp >::operator~ () const
- void std::valarray< _Tp >::resize (size_t __size, _Tp __c=_Tp())
- valarray< _Tp > std::valarray< _Tp >::shift (int __n) const
- size_t std::slice::size () const
- valarray< size_t > std::gslice::size () const
- size_t std::valarray< _Tp >::size () const
- size_t std::slice::start () const
- size_t std::gslice::start () const
- size_t std::slice::stride () const
- valarray< size_t > std::gslice::stride () const
- _Tp std::valarray< _Tp >::sum () const
- void std::valarray< _Tp >::swap (valarray< _Tp > &__v) noexcept

2.23.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

2.23.2 Function Documentation

2.23.2.1 std::gslice::gslice ( ) `[inline]`

Construct an empty slice.

Definition at line 149 of file gslice.h.

2.23.2.2 std::gslice::gslice (size_t *__o*, const valarray< size_t > & *__l*, const valarray< size_t > & *__s*) `[inline]`

Construct a slice.

Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

Parameters

__o	Offset in array of first element.
__l	Array of dimension lengths.
__s	Array of dimension strides between array elements.

Definition at line 153 of file gslice.h.

2.23.2.3 std::gslice::gslice (const gslice & *__g*) `[inline]`

Copy constructor.

Definition at line 158 of file gslice.h.

2.23.2.4 template<typename _Tp > gslice_array< _Tp >::gslice_array (const gslice_array< _Tp > & *__a*) `[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file gslice_array.h.

2.23.2.5 template<typename _Tp > indirect_array< _Tp >::indirect_array (const indirect_array< _Tp > & *__a*) `[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file indirect_array.h.

2.23.2.6 template<typename _Tp > mask_array< _Tp >::mask_array (const mask_array< _Tp > & *a*) `[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 139 of file mask_array.h.

2.23.2.7 std::slice::slice ( ) `[inline]`

Construct an empty slice.

Definition at line 90 of file slice_array.h.

2.23.2.8 std::slice::slice (size_t *__o*, size_t *__d*, size_t *__s*) `[inline]`

Construct a slice.

Parameters

__o	Offset in array of first element.
__d	Number of elements in slice.
__s	Stride between array elements.

Definition at line 94 of file slice_array.h.

2.23.2.9 template<typename _Tp > slice_array< _Tp >::slice_array (const slice_array< _Tp > & *a*) `[inline]`

Copy constructor. Both slices refer to the same underlying array.

Definition at line 207 of file slice_array.h.

2.23.2.10 template<typename _Tp > valarray< _Tp >::valarray ( ) `[inline]`

Construct an empty array.

Definition at line 605 of file valarray.

2.23.2.11 template<typename _Tp > valarray< _Tp >::valarray (size_t *__n*) [inline], [explicit]

Construct an array with *n* elements.

Definition at line 609 of file valarray.

2.23.2.12 template<typename _Tp> valarray< _Tp >::valarray (const _Tp & *__t*, size_t *__n*) [inline]

Construct an array with *n* elements initialized to *t*.

Definition at line 615 of file valarray.

2.23.2.13 template<typename _Tp> valarray< _Tp >::valarray (const valarray< _Tp > & *__v*) [inline]

Copy constructor.

Definition at line 630 of file valarray.

2.23.2.14 template<typename _Tp> valarray< _Tp >::valarray (valarray< _Tp > && *__v*) [inline], [noexcept]

Move constructor.

Definition at line 638 of file valarray.

2.23.2.15 template<typename _Tp> valarray< _Tp >::valarray (const slice_array< _Tp > & *__sa*) [inline]

Construct an array with the same size and values in *sa*.

Definition at line 648 of file valarray.

2.23.2.16 template<typename _Tp> valarray< _Tp >::valarray (const gslice_array< _Tp > & *__ga*) [inline]

Construct an array with the same size and values in *ga*.

Definition at line 657 of file valarray.

2.23.2.17 template<typename _Tp> valarray< _Tp >::valarray (const mask_array< _Tp > & *__ma*) [inline]

Construct an array with the same size and values in *ma*.

Definition at line 668 of file valarray.

2.23.2.18 template<typename _Tp> valarray< _Tp >::valarray (const indirect_array< _Tp > & *__ia*) [inline]

Construct an array with the same size and values in *ia*.

Definition at line 677 of file valarray.

2.23.2.19 template<typename _Tp> valarray< _Tp >::valarray (initializer_list< _Tp > *__l*) [inline]

Construct an array with an initializer_list of values.

Definition at line 687 of file valarray.

2.23.2.20 std::gslice::~gslice ( ) [inline]

Destructor.

Definition at line 163 of file gslice.h.

2.23.2.21 template<class _Tp> _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::apply (_Tp *func_Tp*) const [inline]

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

func	Function of Tp returning Tp to apply.

Returns

New valarray with transformed elements.

Definition at line 1036 of file valarray.

2.23.2.22 template<class _Tp> _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::apply (_Tp *funcconst _Tp &*) const [inline]

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

func	Function of const Tp& returning Tp to apply.

Returns

New valarray with transformed elements.

Definition at line 1044 of file valarray.

2.23.2.23 template<class _Tp > _Tp* std::begin (valarray< _Tp > & *__va*) [inline]

Return an iterator pointing to the first element of the valarray.

Parameters

__va	valarray.

Definition at line 1183 of file valarray.

References std::__addressof().

2.23.2.24 template<class _Tp > const _Tp* std::begin (const valarray< _Tp > & *__va*) [inline]

Return an iterator pointing to the first element of the const valarray.

Parameters

__va	valarray.

Definition at line 1193 of file valarray.

References std::__addressof().

2.23.2.25 template< class _Tp > valarray< _Tp > valarray< _Tp >::cshift (int __n) const [inline]

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i, the new position is (i - n) % size(). The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

__n	Number of element positions to rotate.

Returns

New valarray with elements in shifted positions.

Definition at line 962 of file valarray.

2.23.2.26 template< class _Tp > _Tp* std::end (valarray< _Tp > & __va) [inline]

Return an iterator pointing to one past the last element of the valarray.

Parameters

__va	valarray.

Definition at line 1203 of file valarray.

References std::__addressof(), and std::valarray< _Tp >::size().

2.23.2.27 template< class _Tp > const _Tp* std::end (const valarray< _Tp > & __va) [inline]

Return an iterator pointing to one past the last element of the const valarray.

Parameters

__va	valarray.

Definition at line 1213 of file valarray.

References std::__addressof(), and std::valarray< _Tp >::size().

2.23.2.28 template< typename _Tp > _Tp valarray< _Tp >::max ( ) const [inline]

Return the maximum element using operator<().

Definition at line 1028 of file valarray.

References std::max_element().

2.23.2.29 template< typename _Tp > _Tp valarray< _Tp >::min ( ) const [inline]

Return the minimum element using operator<().

Definition at line 1020 of file valarray.

References std::min_element().

2.23.2.30 template<typename _Tp > valarray< _Tp >::template _UnaryOp< __logical_not >::_Rt valarray< _Tp >::operator! ( ) const [inline]

Return a new valarray by applying unary ! to each element.

Definition at line 1063 of file valarray.

2.23.2.31 template<typename _Tp > void gslice_array< _Tp >::operator%= (const valarray< _Tp > & __v) const [inline]

Modulo slice elements by corresponding elements of *v*.

Definition at line 202 of file gslice_array.h.

2.23.2.32 template<typename _Tp > void mask_array< _Tp >::operator%= (const valarray< _Tp > & __v) const [inline]

Modulo slice elements by corresponding elements of *v*.

Definition at line 192 of file mask_array.h.

2.23.2.33 template<typename _Tp > void indirect_array< _Tp >::operator%= (const valarray< _Tp > & __v) const [inline]

Modulo slice elements by corresponding elements of *v*.

Definition at line 196 of file indirect_array.h.

2.23.2.34 template<typename _Tp > void slice_array< _Tp >::operator%= (const valarray< _Tp > & __v) const [inline]

Modulo slice elements by corresponding elements of *v*.

Definition at line 258 of file slice_array.h.

2.23.2.35 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator%= (const _Tp & __t) [inline]

Set each element e of array to e % *t*.

Definition at line 1090 of file valarray.

2.23.2.36 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator%= (const valarray< _Tp > & __v) [inline]

Modulo elements of array by corresponding elements of *v*.

Definition at line 1090 of file valarray.

2.23.2.37 template<typename _Tp > void gslice_array< _Tp >::operator&= (const valarray< _Tp > & __v) const [inline]

Logical and slice elements with corresponding elements of *v*.

Definition at line 206 of file gslice_array.h.

2.23.2.38 template<typename _Tp > void mask_array< _Tp >::operator&= (const valarray< _Tp > & __v) const [inline]

Logical and slice elements with corresponding elements of *v*.

Definition at line 196 of file mask_array.h.

2.23.2.39 template<typename _Tp > void indirect_array< _Tp >::operator&= (const valarray< _Tp > & __v) const [inline]

Logical and slice elements with corresponding elements of *v*.

Definition at line 200 of file indirect_array.h.

2.23.2.40 template<typename _Tp > void slice_array< _Tp >::operator&= (const valarray< _Tp > & __v) const [inline]

Logical and slice elements with corresponding elements of *v*.

Definition at line 262 of file slice_array.h.

2.23.2.41 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator&= (const _Tp & __t) [inline]

Set each element e of array to e & *t*.

Definition at line 1092 of file valarray.

2.23.2.42 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator&= (const valarray< _Tp > & __v) [inline]

Logical and corresponding elements of *v* with elements of array.

Definition at line 1092 of file valarray.

2.23.2.43 template<typename _Tp > void gslice_array< _Tp >::operator*= (const valarray< _Tp > & __v) const [inline]

Multiply slice elements by corresponding elements of *v*.

Definition at line 200 of file gslice_array.h.

2.23.2.44 template<typename _Tp > void mask_array< _Tp >::operator*= (const valarray< _Tp > & __v) const [inline]

Multiply slice elements by corresponding elements of *v*.

Definition at line 190 of file mask_array.h.

2.23.2.45 template<typename _Tp > void indirect_array< _Tp >::operator*= (const valarray< _Tp > & __v) const [inline]

Multiply slice elements by corresponding elements of *v*.

Definition at line 194 of file indirect_array.h.

2.23.2.46 template<typename _Tp > void slice_array< _Tp >::operator*= (const valarray< _Tp > & __v) const [inline]

Multiply slice elements by corresponding elements of *v*.

Definition at line 256 of file slice_array.h.

2.23.2.47 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator*= (const _Tp & __t) [inline]

Multiply each element of array by *t*.

Definition at line 1088 of file valarray.

2.23.2.48 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator*= (const valarray< _Tp > & __v) [inline]

Multiply elements of array by corresponding elements of *v*.

Definition at line 1088 of file valarray.

2.23.2.49 template<typename _Tp > valarray< _Tp >::template _UnaryOp< __unary_plus >::_Rt valarray< _Tp >::operator+ ( ) const [inline]

Return a new valarray by applying unary + to each element.

Definition at line 1060 of file valarray.

2.23.2.50 template<typename _Tp > void gslice_array< _Tp >::operator+= (const valarray< _Tp > & __v) const [inline]

Add corresponding elements of *v* to slice elements.

Definition at line 203 of file gslice_array.h.

2.23.2.51 template<typename _Tp > void mask_array< _Tp >::operator+= (const valarray< _Tp > & __v) const [inline]

Add corresponding elements of *v* to slice elements.

Definition at line 193 of file mask_array.h.

2.23.2.52 template<typename _Tp > void indirect_array< _Tp >::operator+= (const valarray< _Tp > & __v) const [inline]

Add corresponding elements of *v* to slice elements.

Definition at line 197 of file indirect_array.h.

2.23.2.53 template<typename _Tp > void slice_array< _Tp >::operator+= (const valarray< _Tp > & __v) const [inline]

Add corresponding elements of *v* to slice elements.

Definition at line 259 of file slice_array.h.

2.23.2.54 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator+= (const _Tp & __t) [inline]

Add *t* to each element of array.

Definition at line 1086 of file valarray.

2.23.2.55 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator+= (const valarray< _Tp > & __v) [inline]

Add corresponding elements of *v* to elements of array.

Definition at line 1086 of file valarray.

2.23.2.56 template<typename _Tp > valarray< _Tp >::template _UnaryOp< __negate >::_Rt valarray< _Tp >::operator- ( ) const [inline]

Return a new valarray by applying unary - to each element.

Definition at line 1061 of file valarray.

2.23.2.57 template<typename _Tp > void gslice_array< _Tp >::operator-= (const valarray< _Tp > & __v) const [inline]

Subtract corresponding elements of *v* from slice elements.

Definition at line 204 of file gslice_array.h.

2.23.2.58 template<typename _Tp > void mask_array< _Tp >::operator-= (const valarray< _Tp > & __v) const [inline]

Subtract corresponding elements of *v* from slice elements.

Definition at line 194 of file mask_array.h.

2.23.2.59 template<typename _Tp > void indirect_array< _Tp >::operator-= (const valarray< _Tp > & __v) const [inline]

Subtract corresponding elements of *v* from slice elements.

Definition at line 198 of file indirect_array.h.

2.23.2.60 template<typename _Tp > void slice_array< _Tp >::operator-= (const valarray< _Tp > & __v) const [inline]

Subtract corresponding elements of *v* from slice elements.

Definition at line 260 of file slice_array.h.

2.23.2.61 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator-= (const _Tp & __t) [inline]

Subtract *t* to each element of array.

Definition at line 1087 of file valarray.

2.23.2.62 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator-= (const valarray< _Tp > & __v) [inline]

Subtract corresponding elements of *v* from elements of array.

Definition at line 1087 of file valarray.

2.23.2.63 template<typename _Tp > void gslice_array< _Tp >::operator/= (const valarray< _Tp > & __v) const [inline]

Divide slice elements by corresponding elements of *v*.

Definition at line 201 of file gslice_array.h.

2.23.2.64 template<typename _Tp > void mask_array< _Tp >::operator/= (const valarray< _Tp > & __v) const [inline]

Divide slice elements by corresponding elements of *v*.

Definition at line 191 of file mask_array.h.

2.23.2.65 template<typename _Tp > void indirect_array< _Tp >::operator/= (const valarray< _Tp > & __v) const [inline]

Divide slice elements by corresponding elements of *v*.

Definition at line 195 of file indirect_array.h.

2.23.2.66 template<typename _Tp > void slice_array< _Tp >::operator/= (const valarray< _Tp > & __v) const [inline]

Divide slice elements by corresponding elements of *v*.

Definition at line 257 of file slice_array.h.

2.23.2.67 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator/= (const _Tp & __t) [inline]

Divide each element of array by *t*.

Definition at line 1089 of file valarray.

2.23.2.68 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator/= (const valarray< _Tp > & __v) [inline]

Divide elements of array by corresponding elements of *v*.

Definition at line 1089 of file valarray.

2.23.2.69 template<typename _Tp > void gslice_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const [inline]

Left shift slice elements by corresponding elements of *v*.

Definition at line 208 of file gslice_array.h.

2.23.2.70 template<typename _Tp > void mask_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const [inline]

Left shift slice elements by corresponding elements of *v*.

Definition at line 198 of file mask_array.h.

2.23.2.71 template<typename _Tp > void indirect_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const [inline]

Left shift slice elements by corresponding elements of *v*.

Definition at line 202 of file indirect_array.h.

2.23.2.72 template<typename _Tp > void slice_array< _Tp >::operator<<= (const valarray< _Tp > & __v) const [inline]

Left shift slice elements by corresponding elements of *v*.

Definition at line 264 of file slice_array.h.

2.23.2.73 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator<<= (const _Tp & __t) [inline]

Left shift each element e of array by *t* bits.

Definition at line 1094 of file valarray.

2.23.2.74 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator<<= (const valarray< _Tp > & __v) [inline]

Left shift elements of array by corresponding elements of *v*.

Definition at line 1094 of file valarray.

2.23.2.75 template<typename _Tp > gslice_array< _Tp > & gslice_array< _Tp >::operator= (const gslice_array< _Tp > & __a) [inline]

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 148 of file gslice_array.h.

2.23.2.76 template<typename _Tp > indirect_array< _Tp > & indirect_array< _Tp >::operator= (const indirect_array< _Tp > & __a) [inline]

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file indirect_array.h.

2.23.2.77 template<typename _Tp > mask_array< _Tp > & mask_array< _Tp >::operator= (const mask_array< _Tp > & __a) [inline]

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 149 of file mask_array.h.

2.23.2.78 template<typename _Tp > void gslice_array< _Tp >::operator= (const valarray< _Tp > & __v) const [inline]

Assign slice elements to corresponding elements of *v*.

Definition at line 166 of file gslice_array.h.

References std::valarray< _Tp >::size().

2.23.2.79 template<typename _Tp > void indirect_array< _Tp >::operator= (const valarray< _Tp > & __v) const [inline]

Assign slice elements to corresponding elements of *v*.

Definition at line 168 of file indirect_array.h.

2.23.2.80 gslice & std::gslice::operator= (const gslice & __g) [inline]

Assignment operator.

Definition at line 170 of file gslice.h.

2.23.2.81 template<typename _Tp > void gslice_array< _Tp >::operator= (const _Tp & __t) const [inline]

Assign all slice elements to *t*.

Definition at line 158 of file gslice_array.h.

2.23.2.82 template<typename _Tp > void mask_array< _Tp >::operator= (const _Tp & __t) const [inline]

Assign all slice elements to *t*.

Definition at line 158 of file mask_array.h.

2.23.2.83 template<typename _Tp > void indirect_array< _Tp >::operator= (const _Tp & __t) const [inline]

Assign all slice elements to *t*.

Definition at line 163 of file indirect_array.h.

2.23.2.84 template<typename _Tp > slice_array< _Tp > & slice_array< _Tp >::operator= (const slice_array< _Tp > & __a) [inline]

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 215 of file slice_array.h.

2.23.2.85 template<typename _Tp > void slice_array< _Tp >::operator= (const valarray< _Tp > & __v) const [inline]

Assign slice elements to corresponding elements of *v*.

Definition at line 229 of file slice_array.h.

2.23.2.86 template<typename _Tp > void slice_array< _Tp >::operator= (const _Tp & __t) const [inline]

Assign all slice elements to *t*.

Definition at line 224 of file slice_array.h.

2.23.2.87 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (const valarray< _Tp > & __v) [inline]

Assign elements to an array.

Assign elements of array to values in *v*.

Parameters

__v	Valarray to get values from.

Definition at line 708 of file valarray.

2.23.2.88 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (valarray< _Tp > && __v) [inline], [noexcept]

Move assign elements to an array.

Move assign elements of array to values in *v*.

Parameters

__v	Valarray to get values from.

Definition at line 732 of file valarray.

2.23.2.89 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (const _Tp & __t) [inline]

Assign elements to a value.

Assign all elements of array to *t*.

Parameters

__t	Value for elements.

Definition at line 772 of file valarray.

2.23.2.90 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (const slice_array< _Tp > & __sa) [inline]

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

__sa	Array slice to get values from.

Definition at line 780 of file valarray.

2.23.2.91 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (const gslice_array< _Tp > & __ga) [inline]

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

__ga	Array slice to get values from.

Definition at line 790 of file valarray.

References std::valarray< _Tp >::size().

2.23.2.92 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (const mask_array< _Tp > & __ma) [inline]

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

__ma	Array slice to get values from.

Definition at line 800 of file valarray.

2.23.2.93 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (const indirect_array< _Tp > & __ia) [inline]

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

__ia	Array slice to get values from.

Definition at line 810 of file valarray.

2.23.2.94 template<typename _Tp> valarray< _Tp > & valarray< _Tp >::operator= (initializer_list< _Tp > __l) [inline]

Assign elements to an initializer_list.

Assign elements of array to values in *__l*. Results are undefined if *__l* does not have the same size as this array.

Parameters

__l	initializer_list to get values from.

Definition at line 748 of file valarray.

2.23.2.95 template<typename _Tp > void gslice_array< _Tp >::operator>>= (const valarray< _Tp > & __v) const [inline]

Right shift slice elements by corresponding elements of *v*.

Definition at line 209 of file gslice_array.h.

2.23.2.96 template<typename _Tp > void mask_array< _Tp >::operator>>= (const valarray< _Tp > & __v) const [inline]

Right shift slice elements by corresponding elements of *v*.

Definition at line 199 of file mask_array.h.

2.23.2.97 template<typename _Tp > void indirect_array< _Tp >::operator>>= (const valarray< _Tp > & __v) const [inline]

Right shift slice elements by corresponding elements of *v*.

Definition at line 203 of file indirect_array.h.

2.23.2.98 template< typename _Tp > void slice_array< _Tp >::operator>>= (const valarray< _Tp > & __v) const [inline]

Right shift slice elements by corresponding elements of *v*.

Definition at line 265 of file slice_array.h.

2.23.2.99 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator>>= (const _Tp & __t) [inline]

Right shift each element e of array by *t* bits.

Definition at line 1095 of file valarray.

2.23.2.100 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator>>= (const valarray< _Tp > & __v) [inline]

Right shift elements of array by corresponding elements of *v*.

Definition at line 1095 of file valarray.

2.23.2.101 template< typename _Tp > _Tp & valarray< _Tp >::operator[] (size_t __i) [inline]

Return a reference to the i'th array element.

Parameters

__i	Index of element to return.

Returns

Reference to the i'th element.

Definition at line 576 of file valarray.

2.23.2.102 template< typename _Tp > _Expr< _SClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::operator[] (slice __s) const [inline]

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See Also

slice.

Parameters

__s	The source slice.

Returns

New valarray containing elements in *__s*.

Definition at line 829 of file valarray.

2.23.2.103 template< typename _Tp > slice_array< _Tp > valarray< _Tp >::operator[] (slice __s) [inline]

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See Also

slice.

Parameters

__s	The source slice.

Returns

New valarray containing elements in *__s*.

Definition at line 837 of file valarray.

2.23.2.104 template< typename _Tp > _Expr< _GClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::operator[] (const gslice & __s) const [inline]

Return an array subset.

Returns a slice_array referencing the elements of the array indicated by the slice argument.

See Also

gslice.

Parameters

__s	The source slice.

Returns

Slice_array referencing elements indicated by *__s*.

Definition at line 842 of file valarray.

2.23.2.105 template< typename _Tp > gslice_array< _Tp > valarray< _Tp >::operator[] (const gslice & __s) [inline]

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

See Also

gslice.

Parameters

__s	The source gslice.

Returns

New valarray containing elements in *__s*.

Definition at line 851 of file valarray.

2.23.2.106 template<typename _Tp > valarray< _Tp > valarray< _Tp >::operator[] (const valarray< bool > & *__m*) const [inline]

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

__m	The valarray bitmask.

Returns

New valarray containing elements indicated by *__m*.

Definition at line 859 of file valarray.

References std::valarray< _Tp >::size().

2.23.2.107 template<typename _Tp > mask_array< _Tp > valarray< _Tp >::operator[] (const valarray< bool > & *__m*) [inline]

Return a reference to an array subset.

Returns a new mask_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

__m	The valarray bitmask.

Returns

New valarray containing elements indicated by *__m*.

Definition at line 871 of file valarray.

References std::valarray< _Tp >::size().

2.23.2.108 template<typename _Tp > _Expr< _IClos< _ValArray, _Tp >, _Tp > valarray< _Tp >::operator[] (const valarray< size_t > & *__i*) const [inline]

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

__i	The valarray element index list.

Returns

New valarray containing elements in *__s*.

Definition at line 882 of file valarray.

2.23.2.109 template< typename _Tp > indirect_array< _Tp > valarray< _Tp >::operator[] (const valarray< size_t > & *__i*) [inline]

Return a reference to an array subset.

Returns an indirect_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect_array refers to these elements.

Parameters

__i	The valarray element index list.

Returns

Indirect_array referencing elements in *__i*.

Definition at line 890 of file valarray.

References std::valarray< _Tp >::size().

2.23.2.110 template< typename _Tp > void gslice_array< _Tp >::operator^= (const valarray< _Tp > & *__v*) const [inline]

Logical xor slice elements with corresponding elements of *v*.

Definition at line 205 of file gslice_array.h.

2.23.2.111 template< typename _Tp > void mask_array< _Tp >::operator^= (const valarray< _Tp > & *__v*) const [inline]

Logical xor slice elements with corresponding elements of *v*.

Definition at line 195 of file mask_array.h.

2.23.2.112 template< typename _Tp > void indirect_array< _Tp >::operator^= (const valarray< _Tp > & *__v*) const [inline]

Logical xor slice elements with corresponding elements of *v*.

Definition at line 199 of file indirect_array.h.

2.23.2.113 template< typename _Tp > void slice_array< _Tp >::operator^= (const valarray< _Tp > & *__v*) const [inline]

Logical xor slice elements with corresponding elements of *v*.

Definition at line 261 of file slice_array.h.

2.23.2.114 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator^= (const _Tp & __t) `[inline]`

Set each element e of array to e ^ *t*.

Definition at line 1091 of file valarray.

2.23.2.115 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator^= (const valarray< _Tp > & __v) `[inline]`

Logical xor corresponding elements of *v* with elements of array.

Definition at line 1091 of file valarray.

2.23.2.116 template<typename _Tp > void gslice_array< _Tp >::operator|= (const valarray< _Tp > & __v) const `[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 207 of file gslice_array.h.

2.23.2.117 template<typename _Tp > void mask_array< _Tp >::operator|= (const valarray< _Tp > & __v) const `[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 197 of file mask_array.h.

2.23.2.118 template<typename _Tp > void indirect_array< _Tp >::operator|= (const valarray< _Tp > & __v) const `[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 201 of file indirect_array.h.

2.23.2.119 template<typename _Tp > void slice_array< _Tp >::operator|= (const valarray< _Tp > & __v) const `[inline]`

Logical or slice elements with corresponding elements of *v*.

Definition at line 263 of file slice_array.h.

2.23.2.120 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator|= (const _Tp & __t) `[inline]`

Set each element e of array to e | *t*.

Definition at line 1093 of file valarray.

2.23.2.121 template<class _Tp> valarray< _Tp > & valarray< _Tp >::operator|= (const valarray< _Tp > & __v) `[inline]`

Logical or corresponding elements of *v* with elements of array.

Definition at line 1093 of file valarray.

2.23.2.122 template<typename _Tp > valarray< _Tp >::template _UnaryOp< __bitwise_not >::_Rt valarray< _Tp >::operator∼ ( ) const `[inline]`

Return a new valarray by applying unary ∼ to each element.

Definition at line 1062 of file valarray.

2.23.2.123 template<class _Tp> void valarray< _Tp >::resize (size_t __size, _Tp __c = `_Tp()`) `[inline]`

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

Parameters

__size	New array size.
__c	New value for all elements.

Definition at line 1003 of file valarray.

2.23.2.124 template<class _Tp > valarray< _Tp > valarray< _Tp >::shift (int __n) const [inline]

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i, the new position is i - n. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, n). Negative arguments discard elements from the top of the array.

Parameters

__n	Number of element positions to shift.

Returns

New valarray with elements in shifted positions.

Definition at line 921 of file valarray.

2.23.2.125 size_t std::slice::size ( ) const [inline]

Return size of slice.

Definition at line 102 of file slice_array.h.

2.23.2.126 valarray< size_t > std::gslice::size ( ) const [inline]

Return array of sizes of slice dimensions.

Definition at line 139 of file gslice.h.

2.23.2.127 template<class _Tp > size_t valarray< _Tp >::size ( ) const [inline]

Return the number of elements in array.

Definition at line 908 of file valarray.

Referenced by std::end(), std::gslice_array< _Tp >::operator=(), std::valarray< _Tp >::operator=(), and std::valarray< _Tp >::operator[]().

2.23.2.128 size_t std::slice::start ( ) const [inline]

Return array offset of first slice element.

Definition at line 98 of file slice_array.h.

2.23.2.129 size_t std::gslice::start ( ) const [inline]

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

2.23.2.130 size_t std::slice::stride ( ) const [inline]

Return array stride of slice.

Definition at line 106 of file slice_array.h.

2.23.2.131 valarray< size_t > std::gslice::stride ( ) const [inline]

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.

2.23.2.132 template<class _Tp > _Tp valarray< _Tp >::sum ( ) const [inline]

Return the sum of all elements in the array.

Accumulates the sum of all elements into a Tp using +=. The order of adding the elements is unspecified.

Definition at line 913 of file valarray.

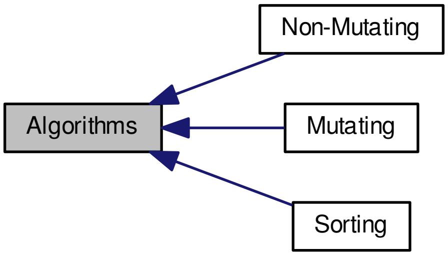
2.23.2.133 template<class _Tp> void valarray< _Tp >::swap (valarray< _Tp > & __v) [inline], [noexcept]

Swap.

Definition at line 899 of file valarray.

2.24 Algorithms

Collaboration diagram for Algorithms:



Modules

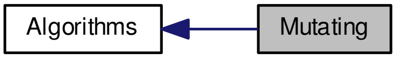
- Mutating
- Non-Mutating
- Sorting

2.24.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

2.25 Mutating

Collaboration diagram for Mutating:



Functions

- template<typename _II , typename _OI >
  _OI std::copy (_II __first, _II __last, _OI __result)
- template<typename _BI1 , typename _BI2 >
  _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
  _OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  _OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _ForwardIterator , typename _Tp >
  void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI , typename _Size , typename _Tp >
  _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _ForwardIterator , typename _Generator >
  void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)
- template<typename _InputIterator , typename _Predicate >
  bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _II , typename _OI >
  _OI std::move (_II __first, _II __last, _OI __result)
- template<typename _BI1 , typename _BI2 >
  _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate >
  pair< _OutputIterator1,
  _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator >
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)

- template<typename _RandomAccessIterator , typename _RandomNumberGenerator >
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
  _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator , typename _Tp >
  void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp >
  _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _ForwardIterator , typename _Predicate , typename _Tp >
  void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- template<typename _BidirectionalIterator >
  void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _OutputIterator >
  _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- template<typename _ForwardIterator >
  void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _OutputIterator >
  _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- template<typename _RandomAccessIterator , typename _UniformRandomNumberGenerator >
  void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _ForwardIterator >
  _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)

- template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)

2.25.1 Detailed Description

2.25.2 Function Documentation

2.25.2.1 template<typename _II , typename _OI > _OI std::copy (_II *__first,* _II *__last,* _OI *__result*) [inline]

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the copy_backward function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 450 of file stl_algobase.h.

2.25.2.2 template<typename _BI1 , typename _BI2 > _BI2 std::copy_backward (_BI1 *__first,* _BI1 *__last,* _BI2 *__result*) [inline]

Copies the range [first,last) into result.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.
__result	A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as copy, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range [first,last). Use copy instead. Note that the start of the output range may overlap [first,last).

Definition at line 619 of file stl_algobase.h.

2.25.2.3 template<typename _InputIterator , typename _OutputIterator , typename _Predicate > _OutputIterator std::copy_if (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* _Predicate *__pred*)

Copy the elements of a sequence for which a predicate is true.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__pred	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [__first,__last) for which __`pred` returns true to the range beginning at __`result`.

copy_if() is stable, so the relative order of elements that are copied is unchanged.

Definition at line 974 of file stl_algo.h.

2.25.2.4 template<typename _InputIterator , typename _Size , typename _OutputIterator > _OutputIterator std::copy_n (_InputIterator *__first,* _Size *__n,* _OutputIterator *__result*) `[inline]`

Copies the range [first,first+n) into [result,result+n).

Parameters

__first	An input iterator.
__n	The number of elements to copy.
__result	An output iterator.

Returns

result+n.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 1037 of file stl_algo.h.

References std::__iterator_category().

2.25.2.5 template<typename _ForwardIterator , typename _Tp > void std::fill (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__value*) `[inline]`

Fills the range [first,last) with copies of value.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__value	A reference-to-const of arbitrary type.

Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 721 of file stl_algobase.h.

2.25.2.6 template<typename _OI , typename _Size , typename _Tp > _OI std::fill_n (_OI *__first,* _Size *__n,* const _Tp & *__value*) `[inline]`

Fills the range [first,first+n) with copies of value.

Parameters

__first	An output iterator.
__n	The count of copies to perform.
__value	A reference-to-const of arbitrary type.

Returns

The iterator at first+n.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or @ wmemset.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 865. More algorithms that throw away information

Definition at line 781 of file stl_algobase.h.

2.25.2.7 template<typename _ForwardIterator , typename _Generator > void std::generate (_ForwardIterator *__first,* _ForwardIterator *__last,* _Generator *__gen*)

Assign the result of a function object to each value in a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__gen	A function object taking no arguments and returning std::iterator_traits<_ForwardIterator>::value_type

Returns

generate() returns no value.

Performs the assignment `*i` = `__gen()` for each `i` in the range [__first,__last).

Definition at line 5071 of file stl_algo.h.

2.25.2.8 template<typename _OutputIterator , typename _Size , typename _Generator > _OutputIterator std::generate_n (_OutputIterator *__first,* _Size *__n,* _Generator *__gen*)

Assign the result of a function object to each value in a sequence.

Parameters

__first	A forward iterator.
__n	The length of the sequence.
__gen	A function object taking no arguments and returning std::iterator_traits<_ForwardIterator>::value_type

Returns

The end of the sequence, __`first+__n`

Performs the assignment `*i` = __`gen()` for each `i` in the range [__first,__first+__n).

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 865. More algorithms that throw away information

Definition at line 5102 of file stl_algo.h.

2.25.2.9 template<typename _InputIterator , typename _Predicate > bool std::is_partitioned (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred*) `[inline]`

Checks whether the sequence is partitioned.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

True if the range [__first,__last) is partioned by __`pred`, i.e. if all elements that satisfy __`pred` appear before those that do not.

Definition at line 826 of file stl_algo.h.

References std::find_if_not(), and std::none_of().

2.25.2.10 template<typename _ForwardIterator1 , typename _ForwardIterator2 > void std::iter_swap (_ForwardIterator1 *__a,* _ForwardIterator2 *__b*) `[inline]`

Swaps the contents of two iterators.

Parameters

__a	An iterator.
__b	Another iterator.

Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 119 of file stl_algobase.h.

Referenced by std::__merge_without_buffer(), std::__move_median_first(), std::__partition(), std::__reverse(), std::__rotate(), std::__unguarded_partition(), std::next_permutation(), std::prev_permutation(), std::random_shuffle(), std::shuffle(), and std::swap_ranges().

2.25.2.11 template<typename _II , typename _OI > _OI std::move (_II __first, _II __last, _OI __result) [inline]

Moves the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

result + (first - last)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the move_backward function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 483 of file stl_algobase.h.

2.25.2.12 template<typename _BI1 , typename _BI2 > _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result) [inline]

Moves the range [first,last) into result.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.
__result	A bidirectional iterator.

Returns

result - (first - last)

The function has the same effect as move, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range (first,last]. Use move instead. Note that the start of the output range may overlap [first,last).

Definition at line 655 of file stl_algobase.h.

2.25.2.13 template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred) [inline]

Move elements for which a predicate is true to the beginning of a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range [__first,middle) and false for each `i` in the range [middle,__last).

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 5274 of file stl_algo.h.

References std::__iterator_category(), and std::__partition().

2.25.2.14 template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate > pair<_OutputIterator1, _OutputIterator2> std::partition_copy (_InputIterator *__first*, _InputIterator *__last*, _OutputIterator1 *__out_true*, _OutputIterator2 *__out_false*, _Predicate *__pred*)

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

Parameters

__first	An input iterator.
__last	An input iterator.
__out_true	An output iterator.
__out_false	An output iterator.
__pred	A predicate.

Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range [__first,__last) for which `__pred` returns true to the range beginning at `out_true` and each element for which `__pred` returns false to `__out_false`.

Definition at line 1066 of file stl_algo.h.

2.25.2.15 template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::partition_point (_ForwardIterator *__first*, _ForwardIterator *__last*, _Predicate *__pred*)

Find the partition point of a partitioned range.

Parameters

__first	An iterator.
__last	Another iterator.
__pred	A predicate.

Returns

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

Definition at line 844 of file stl_algo.h.

References std::advance(), and std::distance().

2.25.2.16 template<typename _RandomAccessIterator > void std::random_shuffle (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) `[inline]`

Randomly shuffle the elements of a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

Nothing.

Reorder the elements in the range [__first,__last) using a random distribution, so that every possible ordering of the sequence is equally likely.

Definition at line 5210 of file stl_algo.h.

References std::iter_swap().

2.25.2.17 template<typename _RandomAccessIterator , typename _RandomNumberGenerator > void std::random_shuffle (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _RandomNumberGenerator && *__rand*)

Shuffle the elements of a sequence using a random number generator.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__rand	The RNG functor or function.

Returns

Nothing.

Reorders the elements in the range [__first,__last) using `__rand` to provide a random distribution. Calling `__rand(N)` for a positive integer `N` should return a randomly chosen integer from the range [0,N).

Definition at line 5238 of file stl_algo.h.

References std::iter_swap().

2.25.2.18 template<typename _ForwardIterator , typename _Tp > _ForwardIterator std::remove (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__value*)

Remove elements from a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__value	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range [__first,__last).

remove() is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1115 of file stl_algo.h.

2.25.2.19 template<typename _InputIterator , typename _OutputIterator , typename _Tp > _OutputIterator std::remove_copy (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* const _Tp & *__value*)

Copy a sequence, removing elements of a given value.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__value	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [__first,__last) not equal to `__value` to the range beginning at `__result`. remove_copy() is stable, so the relative order of elements that are copied is unchanged.

Definition at line 897 of file stl_algo.h.

2.25.2.20 template<typename _InputIterator , typename _OutputIterator , typename _Predicate > _OutputIterator std::remove_copy_if (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* _Predicate *__pred*)

Copy a sequence, removing elements for which a predicate is true.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__pred	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [__first,__last) for which `__pred` returns false to the range beginning at `__result`.

remove_copy_if() is stable, so the relative order of elements that are copied is unchanged.

Definition at line 935 of file stl_algo.h.

2.25.2.21 template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::remove_if (_ForwardIterator *__first,* _ForwardIterator *__last,* _Predicate *__pred*)

Remove elements from a sequence using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range [__first,__last).

remove_if() is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1158 of file stl_algo.h.

2.25.2.22 template<typename _ForwardIterator , typename _Tp > void std::replace (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__old_value,* const _Tp & *__new_value*)

Replace each occurrence of one value in a sequence with another value.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__old_value	The value to be replaced.
__new_value	The replacement value.

Returns

replace() returns no value.

For each iterator `i` in the range [__first,__last) if `*i` == `__old_value` then the assignment `*i` = `__new_value` is performed.

Definition at line 5007 of file stl_algo.h.

2.25.2.23 template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp > _OutputIterator std::replace_copy_if (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* _Predicate *__pred,* const _Tp & *__new_value*)

Copy a sequence, replacing each value for which a predicate returns true with another value.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__pred	A predicate.
__new_value	The replacement value.

Returns

The end of the output sequence, `__result`+(__last-__first).

Copies each element in the range [__first,__last) to the range [__result,__result+(__last-__first)) replacing elements for which `__pred` returns true with `__new_value`.

Definition at line 3945 of file stl_algo.h.

2.25.2.24 template<typename _ForwardIterator , typename _Predicate , typename _Tp > void std::replace_if (_ForwardIterator *__first*, _ForwardIterator *__last*, _Predicate *__pred*, const _Tp & *__new_value*)

Replace each value in a sequence for which a predicate returns true with another value.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate.
__new_value	The replacement value.

Returns

replace_if() returns no value.

For each iterator `i` in the range [__first,__last) if `__pred(*i)` is true then the assignment `*i` = `__new_value` is performed.

Definition at line 5039 of file stl_algo.h.

2.25.2.25 template<typename _BidirectionalIterator > void std::reverse (_BidirectionalIterator *__first*, _BidirectionalIterator *__last*) `[inline]`

Reverse a sequence.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.

Returns

reverse() returns no value.

Reverses the order of the elements in the range [__first,__last), so that the first element becomes the last etc. For every `i` such that `0<=i<=`(__last-__first)/2), `reverse()` swaps *(__first+i) and *(__last-(i+1))

Definition at line 1466 of file stl_algo.h.

References std::__iterator_category(), and std::__reverse().

Referenced by std::next_permutation(), and std::prev_permutation().

2.25.2.26 template<typename _BidirectionalIterator , typename _OutputIterator > _OutputIterator std::reverse_copy (_BidirectionalIterator *__first*, _BidirectionalIterator *__last*, _OutputIterator *__result*)

Copy a sequence, reversing its elements.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.
__result	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range [__first,__last) to the range [__result,__result+(__last-__first)) such that the order of the elements is reversed. For every `i` such that `0<=i<=`(__last-__first), `reverse_copy()` performs the assignment ∗(__result+(__last-__first)-1-i) = ∗(__first+i). The ranges [__first,__last) and [__result,__result+(__last-__first)) must not overlap.

Definition at line 1493 of file stl_algo.h.

2.25.2.27 template<typename _ForwardIterator > void std::rotate (_ForwardIterator *__first,* _ForwardIterator *__middle,* _ForwardIterator *__last*) `[inline]`

Rotate the elements of a sequence.

Parameters

__first	A forward iterator.
__middle	A forward iterator.
__last	A forward iterator.

Returns

Nothing.

Rotates the elements of the range [__first,__last) by (__middle - __first) positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range [__first,__last).

This effectively swaps the ranges [__first,__middle) and [__middle,__last).

Performs ∗(__first+(n+(__last-__middle))%(__last-__first))=∗(__first+n) for each `n` in the range [0,__last-__first).

Definition at line 1699 of file stl_algo.h.

References std::__rotate().

Referenced by std::__inplace_stable_partition(), std::__merge_without_buffer(), std::__rotate_adaptive(), and std::__stable_partition_adaptive().

2.25.2.28 template<typename _ForwardIterator , typename _OutputIterator > _OutputIterator std::rotate_copy (_ForwardIterator *__first,* _ForwardIterator *__middle,* _ForwardIterator *__last,* _OutputIterator *__result*)

Copy a sequence, rotating its elements.

Parameters

__first	A forward iterator.
__middle	A forward iterator.
__last	A forward iterator.
__result	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range [__first,__last) to the range beginning at

Returns

, rotating the copied elements by (__middle-__first) positions so that the element at `__middle` is moved to `__result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range [__first,__last).

Performs ∗(__result+(n+(__last-__middle))%(__last-__first))=∗(__first+n) for each `n` in the range [0,__last-__first).

Definition at line 1735 of file stl_algo.h.

2.25.2.29 template<typename _RandomAccessIterator , typename _UniformRandomNumberGenerator > void std::shuffle (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _UniformRandomNumberGenerator && *__g*)

Shuffle the elements of a sequence using a uniform random number generator.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__g	A UniformRandomNumberGenerator (26.5.1.3).

Returns

Nothing.

Reorders the elements in the range [__first,__last) using `__g` to provide random numbers.

Definition at line 4390 of file stl_algo.h.

References std::iter_swap().

2.25.2.30 template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::stable_partition (_ForwardIterator *__first,* _ForwardIterator *__last,* _Predicate *__pred*)

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range [first,middle) and false for each `i` in the range [middle,last).

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range [__first,__last) such that `__pred(x)==__pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1914 of file stl_algo.h.

References std::__find_if_not(), std::__inplace_stable_partition(), std::__stable_partition_adaptive(), std::_Temporary_buffer< _ForwardIterator, _Tp >::begin(), std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size(), and std::_Temporary_buffer< _ForwardIterator, _Tp >::size().

2.25.2.31 template<typename _ForwardIterator1 , typename _ForwardIterator2 > _ForwardIterator2 std::swap_ranges (_ForwardIterator1 *__first1*, _ForwardIterator1 *__last1*, _ForwardIterator2 *__first2*)

Swap the elements of two sequences.

Parameters

__first1	A forward iterator.
__last1	A forward iterator.
__first2	A forward iterator.

Returns

An iterator equal to `first2+`(last1-first1).

Swaps each element in the range [first1,last1) with the corresponding element in the range [first2,(last1-first1)). The ranges must not overlap.

Definition at line 165 of file stl_algobase.h.

References std::iter_swap().

Referenced by std::__rotate().

2.25.2.32 template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation > _OutputIterator std::transform (_InputIterator *__first*, _InputIterator *__last*, _OutputIterator *__result*, _UnaryOperation *__unary_op*)

Perform an operation on a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__unary_op	A unary operator.

Returns

An output iterator equal to ___`result+`(__last-__first).

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates ∗(__result+N)=unary_op(∗(__first+N)) for each `N` in the range [0,__last-__first).

`unary_op` must not alter its argument.

Definition at line 4938 of file stl_algo.h.

2.25.2.33 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _BinaryOperation > _OutputIterator std::transform (_InputIterator1 *__first1*, _InputIterator1 *__last1*, _InputIterator2 *__first2*, _OutputIterator *__result*, _BinaryOperation *__binary_op*)

Perform an operation on corresponding elements of two sequences.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__result	An output iterator.
__binary_op	A binary operator.

Returns

An output iterator equal to `result+`(last-first).

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates $*$(__result+N)=__binary_op($*$(__first1+N),$*$(__first2+N)) for each `N` in the range [0,__last1-__first1).

`binary_op` must not alter either of its arguments.

Definition at line 4975 of file stl_algo.h.

2.25.2.34 template<typename _ForwardIterator > _ForwardIterator std::unique (_ForwardIterator *__first,* _ForwardIterator *__last*)

Remove consecutive duplicate values from a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. unique() is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1198 of file stl_algo.h.

2.25.2.35 template<typename _ForwardIterator , typename _BinaryPredicate > _ForwardIterator std::unique (_ForwardIterator *__first,* _ForwardIterator *__last,* _BinaryPredicate *__binary_pred*)

Remove consecutive values from a sequence using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__binary_pred	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. unique() is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1238 of file stl_algo.h.

2.25.2.36 template<typename _InputIterator , typename _OutputIterator > _OutputIterator std::unique_copy (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result*) `[inline]`

Copy a sequence, removing consecutive duplicate values.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [__first,__last) to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. unique_copy() is stable, so the relative order of elements that are copied is unchanged.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 241. Does unique_copy() require CopyConstructible and Assignable?

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 538. 241 again: Does unique_copy() require CopyConstructible and Assignable?

Definition at line 5139 of file stl_algo.h.

References std::__iterator_category(), and std::__unique_copy().

2.25.2.37 template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate > _OutputIterator std::unique_copy (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* _BinaryPredicate *__binary_pred*) `[inline]`

Copy a sequence, removing consecutive values using a predicate.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__binary_pred	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range [__first,__last) to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. unique_copy() is stable, so the relative order of elements that are copied is unchanged.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 241. Does unique_copy() require CopyConstructible and Assignable?

Definition at line 5179 of file stl_algo.h.

References std::__iterator_category(), and std::__unique_copy().

2.26 Non-Mutating

Collaboration diagram for Non-Mutating:

Functions

- template<typename _ForwardIterator >
  _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator , typename _Predicate >
  bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Predicate >
  bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Tp >
  iterator_traits
  < _InputIterator >
  ::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _InputIterator , typename _Predicate >
  iterator_traits
  < _InputIterator >
  ::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
  bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _II1 , typename _II2 >
  bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _InputIterator , typename _Tp >
  _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _InputIterator , typename _ForwardIterator >
  _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)
- template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
  _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)

- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Function >
  _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _InputIterator1 , typename _InputIterator2 >
  pair< _InputIterator1,
  _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
  pair< _InputIterator1,
  _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator , typename _Predicate >
  bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _ForwardIterator , typename _Integer , typename _Tp >
  _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)

2.26.1 Detailed Description

2.26.2 Function Documentation

2.26.2.1 template<typename _ForwardIterator > _ForwardIterator std::adjacent_find (_ForwardIterator *__first*, _ForwardIterator *__last*)

Find two adjacent values in a sequence that are equal.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in [__first,__last) and such that `*i` == *(i+1), or `__last` if no such iterator exists.

Definition at line 4581 of file stl_algo.h.

2.26.2.2 template<typename _ForwardIterator , typename _BinaryPredicate > _ForwardIterator std::adjacent_find (_ForwardIterator *__first*, _ForwardIterator *__last*, _BinaryPredicate *__binary_pred*)

Find two adjacent values in a sequence using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__binary_pred	A binary predicate.

Returns

> The first iterator `i` such that `i` and `i+1` are both valid iterators in [__first,__last) and such that `__binary_pred`(∗i,∗(i+1)) is true, or `__last` if no such iterator exists.

Definition at line 4613 of file stl_algo.h.

2.26.2.3 template<typename _InputIterator , typename _Predicate > bool std::all_of (_InputIterator *__first*, _InputIterator *__last*, _Predicate *__pred*) `[inline]`

Checks that a predicate is true for all the elements of a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

> True if the check is true, false otherwise.

Returns true if `__pred` is true for each element in the range [__first,__last), and false otherwise.

Definition at line 753 of file stl_algo.h.

References std::find_if_not().

2.26.2.4 template<typename _InputIterator , typename _Predicate > bool std::any_of (_InputIterator *__first*, _InputIterator *__last*, _Predicate *__pred*) `[inline]`

Checks that a predicate is false for at least an element of a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

> True if the check is true, false otherwise.

Returns true if an element exists in the range [__first,__last) such that `__pred` is true, and false otherwise.

Definition at line 788 of file stl_algo.h.

References std::none_of().

2.26.2.5 template<typename _InputIterator , typename _Tp > iterator_traits<_InputIterator>::difference_type std::count (_InputIterator *__first,* _InputIterator *__last,* const _Tp & *__value*)

Count the number of copies of a value in a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__value	The value to be counted.

Returns

The number of iterators `i` in the range [__first,__last) for which `*i` == __`value`

Definition at line 4645 of file stl_algo.h.

2.26.2.6 template<typename _InputIterator , typename _Predicate > iterator_traits<_InputIterator>::difference_type std::count_if (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred*)

Count the elements of a sequence for which a predicate is true.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

The number of iterators `i` in the range [__first,__last) for which __`pred(*i)` is true.

Definition at line 4670 of file stl_algo.h.

Referenced by std::is_permutation().

2.26.2.7 template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate > bool std::equal (_IIter1 *__first1,* _IIter1 *__last1,* _IIter2 *__first2,* _BinaryPredicate *__binary_pred*) `[inline]`

Tests a range for element-wise equality.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__binary_pred	A binary predicate functor.

Returns

A boolean true or false.

This compares the elements of two ranges using the binary_pred parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1053 of file stl_algobase.h.

2.26.2.8 template<typename _II1 , typename _II2 > bool std::equal (_II1 *__first1,* _II1 *__last1,* _II2 *__first2*) `[inline]`

Tests a range for element-wise equality.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using == and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1021 of file stl_algobase.h.

Referenced by std::operator==().

2.26.2.9 template<typename _InputIterator , typename _Tp > _InputIterator std::find (_InputIterator *__first,* _InputIterator *__last,* const _Tp & *__val*) `[inline]`

Find the first occurrence of a value in a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__val	The value to find.

Returns

The first iterator `i` in the range [__first,__last) such that `*i` == `__val`, or `__last` if no such iterator exists.

Definition at line 4455 of file stl_algo.h.

References std::__find(), and std::__iterator_category().

2.26.2.10 template<typename _ForwardIterator1 , typename _ForwardIterator2 > _ForwardIterator1 std::find_end (_ForwardIterator1 *__first1,* _ForwardIterator1 *__last1,* _ForwardIterator2 *__first2,* _ForwardIterator2 *__last2*) `[inline]`

Find last matching subsequence in a sequence.

Parameters

__first1	Start of range to search.
__last1	End of range to search.
__first2	Start of sequence to match.
__last2	End of sequence to match.

Returns

> The last iterator `i` in the range [__first1,__last1-(__last2-__first2)) such that ∗(i+N) == ∗(__first2+N) for each `N` in the range [0,__last2-__first2), or `__last1` if no such iterator exists.

Searches the range [__first1,__last1) for a sub-sequence that compares equal value-by-value with the sequence given by [__first2,__last2) and returns an iterator to the __first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in [__first,__last1).

Because the sub-sequence must lie completely within the range [__first1,__last1) it must start at a position less than `__last1-(`__last2-__first2) where `__last2-__first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range [__first1,__last1-(__last2-__first2))

Definition at line 671 of file stl_algo.h.

References std::__iterator_category().

2.26.2.11 template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate > _ForwardIterator1 std::find_end (_ForwardIterator1 *__first1,* _ForwardIterator1 *__last1,* _ForwardIterator2 *__first2,* _ForwardIterator2 *__last2,* _BinaryPredicate *__comp*) `[inline]`

Find last matching subsequence in a sequence using a predicate.

Parameters

__first1	Start of range to search.
__last1	End of range to search.
__first2	Start of sequence to match.
__last2	End of sequence to match.
__comp	The predicate to use.

Returns

> The last iterator `i` in the range [__first1,__last1-(__last2-__first2)) such that `predicate`(∗(i+N), (__first2+N)) is true for each `N` in the range [0,__last2-__first2), or `__last1` if no such iterator exists.

Searches the range [__first1,__last1) for a sub-sequence that compares equal value-by-value with the sequence given by [__first2,__last2) using comp as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in [__first,__last1).

Because the sub-sequence must lie completely within the range [__first1,__last1) it must start at a position less than `__last1-(`__last2-__first2) where `__last2-__first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range [__first1,__last1-(__last2-__first2))

Definition at line 719 of file stl_algo.h.

References std::__iterator_category().

2.26.2.12 template<typename _InputIterator , typename _ForwardIterator > _InputIterator std::find_first_of (_InputIterator *__first1,* _InputIterator *__last1,* _ForwardIterator *__first2,* _ForwardIterator *__last2*)

Find element from a set in a sequence.

Parameters

__first1	Start of range to search.
__last1	End of range to search.
__first2	Start of match candidates.
__last2	End of match candidates.

Returns

> The first iterator `i` in the range [__first1,__last1) such that $*$`i` == $*$(i2) such that i2 is an iterator in [__first2,__last2), or `___last1` if no such iterator exists.

Searches the range [__first1,__last1) for an element that is equal to some element in the range [__first2,__last2). If found, returns an iterator in the range [__first1,__last1), otherwise returns `___last1`.

Definition at line 4509 of file stl_algo.h.

2.26.2.13 template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate > _InputIterator std::find_first_of (_InputIterator *__first1,* _InputIterator *__last1,* _ForwardIterator *__first2,* _ForwardIterator *__last2,* _BinaryPredicate *__comp*)

Find element from a set in a sequence using a predicate.

Parameters

__first1	Start of range to search.
__last1	End of range to search.
__first2	Start of match candidates.
__last2	End of match candidates.
__comp	Predicate to use.

Returns

> The first iterator `i` in the range [__first1,__last1) such that `comp(`$*$i, $*$(i2)) is true and i2 is an iterator in [__first2,__last2), or `___last1` if no such iterator exists.

Searches the range [__first1,__last1) for an element that is equal to some element in the range [__first2,__last2). If found, returns an iterator in the range [__first1,__last1), otherwise returns `___last1`.

Definition at line 4550 of file stl_algo.h.

2.26.2.14 template<typename _InputIterator , typename _Predicate > _InputIterator std::find_if (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred*) `[inline]`

Find the first element in a sequence for which a predicate is true.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

> The first iterator `i` in the range [__first,__last) such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 4479 of file stl_algo.h.

References std::__find_if(), and std::__iterator_category().

2.26.2.15 template<typename _InputIterator , typename _Predicate > _InputIterator std::find_if_not (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred*) `[inline]`

Find the first element in a sequence for which a predicate is false.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

> The first iterator `i` in the range [__first,__last) such that __`pred(*i)` is false, or __`last` if no such iterator exists.

Definition at line 803 of file stl_algo.h.

References std::__find_if_not().

Referenced by std::all_of(), and std::is_partitioned().

2.26.2.16 template<typename _InputIterator , typename _Function > _Function std::for_each (_InputIterator *__first*, _InputIterator *__last*, _Function *__f*)

Apply a function to every element of a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__f	A unary function object.

Returns

> __`f` (std::move(__`f`) in C++0x).

Applies the function object __`f` to each element in the range [first,last). __`f` must not modify the order of the sequence. If __`f` has a return value it is ignored.

Definition at line 4434 of file stl_algo.h.

2.26.2.17 template<typename _ForwardIterator1 , typename _ForwardIterator2 > bool std::is_permutation (_ForwardIterator1 *__first1*, _ForwardIterator1 *__last1*, _ForwardIterator2 *__first2*)

Checks whether a permutaion of the second sequence is equal to the first sequence.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.

Returns

> true if there exists a permutation of the elements in the range [__first2, __first2 + (__last1 - __first1)), beginning with ForwardIterator2 begin, such that equal(__first1, __last1, begin) returns true; otherwise, returns false.

Definition at line 4294 of file stl_algo.h.

References std::advance(), std::count(), and std::distance().

2.26.2.18 template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate > bool std::is_permutation (_ForwardIterator1 *__first1,* _ForwardIterator1 *__last1,* _ForwardIterator2 *__first2,* _BinaryPredicate *__pred*)

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__pred	A binary predicate.

Returns

> true if there exists a permutation of the elements in the range [__first2, __first2 + (__last1 - __first1)), beginning with ForwardIterator2 begin, such that equal(__first1, __last1, __begin, __pred) returns true; otherwise, returns false.

Definition at line 4340 of file stl_algo.h.

References std::advance(), std::bind(), std::count_if(), and std::distance().

2.26.2.19 template<typename _InputIterator1 , typename _InputIterator2 > pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2*)

Finds the places in ranges which don't match.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.

Returns

> A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using == and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1160 of file stl_algobase.h.

2.26.2.20 template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate > pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _BinaryPredicate *__binary_pred*)

Finds the places in ranges which don't match.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__binary_pred	A binary predicate functor.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the binary_pred parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1198 of file stl_algobase.h.

2.26.2.21 template<typename _InputIterator , typename _Predicate > bool std::none_of (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred*) `[inline]`

Checks that a predicate is false for all the elements of a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range [__first,__last), and false otherwise.

Definition at line 770 of file stl_algo.h.

Referenced by std::any_of(), and std::is_partitioned().

2.26.2.22 template<typename _ForwardIterator1 , typename _ForwardIterator2 > _ForwardIterator1 std::search (_ForwardIterator1 *__first1,* _ForwardIterator1 *__last1,* _ForwardIterator2 *__first2,* _ForwardIterator2 *__last2*)

Search a sequence for a matching sub-sequence.

Parameters

__first1	A forward iterator.
__last1	A forward iterator.
__first2	A forward iterator.
__last2	A forward iterator.

Returns

The first iterator `i` in the range [__first1,__last1-(__last2-__first2)) such that ∗(i+N) == ∗(__first2+N) for each `N` in the range [0,__last2-__first2), or `__last1` if no such iterator exists.

Searches the range [__first1,__last1) for a sub-sequence that compares equal value-by-value with the sequence given by [__first2,__last2) and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range [__first1,__last1) it must start at a position less than `__last1-(`__last2-__first2) where `__last2-__first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range [__first1,__last1-(__last2-__first2))

Definition at line 4712 of file stl_algo.h.

2.26.2.23 template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate > _ForwardIterator1 std::search (_ForwardIterator1 *__first1,* _ForwardIterator1 *__last1,* _ForwardIterator2 *__first2,* _ForwardIterator2 *__last2,* _BinaryPredicate *__predicate*)

Search a sequence for a matching sub-sequence using a predicate.

Parameters

__first1	A forward iterator.
__last1	A forward iterator.
__first2	A forward iterator.
__last2	A forward iterator.
__predicate	A binary predicate.

Returns

> The first iterator `i` in the range [__first1,__last1-(__last2-__first2)) such that `__predicate`(∗(i+N),∗(__first2+N)) is true for each `N` in the range [0,__last2-__first2), or `__last1` if no such iterator exists.

Searches the range [__first1,__last1) for a sub-sequence that compares equal value-by-value with the sequence given by [__first2,__last2), using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

See Also

> search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)

Definition at line 4784 of file stl_algo.h.

2.26.2.24 template<typename _ForwardIterator , typename _Integer , typename _Tp > _ForwardIterator std::search_n (_ForwardIterator *__first,* _ForwardIterator *__last,* _Integer *__count,* const _Tp & *__val*)

Search a sequence for a number of consecutive values.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__count	The number of consecutive values.
__val	The value to find.

Returns

> The first iterator `i` in the range [__first,__last-__count) such that ∗(i+N) == `__val` for each `N` in the range [0,__count), or `__last` if no such iterator exists.

Searches the range [__first,__last) for `count` consecutive elements equal to `__val`.

Definition at line 4858 of file stl_algo.h.

References std::__iterator_category(), and std::__search_n().

2.26.2.25 template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate > _ForwardIterator std::search_n (_ForwardIterator *__first,* _ForwardIterator *__last,* _Integer *__count,* const _Tp & *__val,* _BinaryPredicate *__binary_pred*)

Search a sequence for a number of consecutive values using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__count	The number of consecutive values.
__val	The value to find.
__binary_pred	A binary predicate.

Returns

> The first iterator `i` in the range [__first,__last-__count) such that __`binary_pred`(∗(i+N),__val) is true for each `N` in the range [0,__count), or __`last` if no such iterator exists.

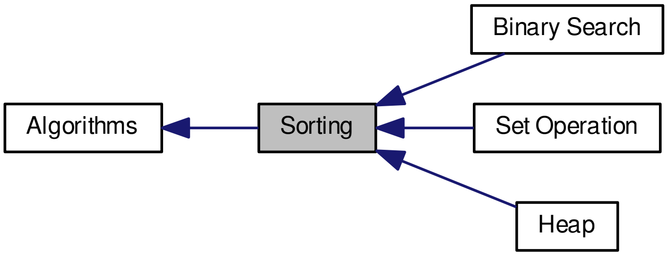
Searches the range [__first,__last) for __`count` consecutive elements for which the predicate returns true.

Definition at line 4896 of file stl_algo.h.

References std::__iterator_category(), and std::__search_n().

2.27 Sorting

Collaboration diagram for Sorting:



Modules

- Binary Search
- Heap
- Set Operation

Functions

- template<typename _BidirectionalIterator >
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _ForwardIterator >
  bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator >
  _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _II1 , typename _II2 >
  bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1 , typename _II2 , typename _Compare >
  bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _Tp >
  const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator >
  _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)

- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp >
  const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator >
  _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp >
  pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator >
  pair< _ForwardIterator,
  _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  pair< _ForwardIterator,
  _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _BidirectionalIterator >
  bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _RandomAccessIterator >
  _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _BidirectionalIterator >
  bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)

- template<typename _RandomAccessIterator >
  void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)

2.27.1 Detailed Description

2.27.2 Function Documentation

2.27.2.1 template<typename _BidirectionalIterator > void std::inplace_merge (_BidirectionalIterator *__first,* _BidirectionalIterator *__middle,* _BidirectionalIterator *__last*)

Merges two sorted ranges in place.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.

Returns

Nothing.

Merges two sorted and consecutive ranges, [__first,__middle) and [__middle,__last), and puts the result in [__first,__last). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (__last-__first)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(__first,__last).

Definition at line 3177 of file stl_algo.h.

References std::__merge_adaptive(), std::__merge_without_buffer(), std::_Temporary_buffer< _ForwardIterator, _Tp >::begin(), std::distance(), and std::_Temporary_buffer< _ForwardIterator, _Tp >::size().

2.27.2.2 template<typename _BidirectionalIterator , typename _Compare > void std::inplace_merge (_BidirectionalIterator *__first,* _BidirectionalIterator *__middle,* _BidirectionalIterator *__last,* _Compare *__comp*)

Merges two sorted ranges in place.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.
__comp	A functor to use for comparisons.

Returns

Nothing.

Merges two sorted and consecutive ranges, [__first,__middle) and [middle,last), and puts the result in [__first,__last). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes (__last-__first)-1 comparisons. Otherwise an NlogN algorithm is used, where N is distance(__first,__last).

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 3232 of file stl_algo.h.

References std::__merge_adaptive(), std::__merge_without_buffer(), std::_Temporary_buffer< _ForwardIterator, _Tp >::begin(), std::distance(), and std::_Temporary_buffer< _ForwardIterator, _Tp >::size().

2.27.2.3 template<typename _ForwardIterator > bool std::is_sorted (_ForwardIterator *__first,* _ForwardIterator *__last*) `[inline]`

Determines whether the elements of a sequence are sorted.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3975 of file stl_algo.h.

References std::is_sorted_until().

2.27.2.4 template<typename _ForwardIterator , typename _Compare > bool std::is_sorted (_ForwardIterator *__first,* _ForwardIterator *__last,* _Compare *__comp*) `[inline]`

Determines whether the elements of a sequence are sorted according to a comparison functor.

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3989 of file stl_algo.h.

References std::is_sorted_until().

2.27.2.5 template<typename _ForwardIterator > _ForwardIterator std::is_sorted_until (_ForwardIterator *__first,* _ForwardIterator *__last*)

Determines the end of a sorted sequence.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

An iterator pointing to the last iterator i in [__first, __last) for which the range [__first, i) is sorted.

Definition at line 4003 of file stl_algo.h.

2.27.2.6 template<typename _ForwardIterator , typename _Compare > _ForwardIterator std::is_sorted_until (_ForwardIterator *__first,* _ForwardIterator *__last,* _Compare *__comp*)

Determines the end of a sorted sequence using comparison functor.

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

An iterator pointing to the last iterator i in [__first, __last) for which the range [__first, i) is sorted.

Definition at line 4032 of file stl_algo.h.

Referenced by std::is_sorted().

2.27.2.7 template<typename _II1 , typename _II2 > bool std::lexicographical_compare (_II1 *__first1,* _II1 *__last1,* _II2 *__first2,* _II2 *__last2*) `[inline]`

Performs **dictionary** comparison on ranges.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.

Returns

A boolean true or false.

Returns true if the sequence of elements defined by the range [first1,last1) is lexicographically less than the sequence of elements defined by the range [first2,last2). Returns false otherwise. (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1084 of file stl_algobase.h.

2.27.2.8 template<typename _II1 , typename _II2 , typename _Compare > bool std::lexicographical_compare (_II1 *__first1,* _II1 *__last1,* _II2 *__first2,* _II2 *__last2,* _Compare *__comp*)

Performs **dictionary** comparison on ranges.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.
__comp	A comparison functor.

Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the comp parameter instead of <.

Definition at line 1120 of file stl_algobase.h.

Referenced by std::operator<().

2.27.2.9 template<typename _Tp > const _Tp & std::max (const _Tp & *__a,* const _Tp & *__b*) `[inline]`

This does what you think it does.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.

Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 216 of file stl_algobase.h.

Referenced by __gnu_parallel::__parallel_nth_element(), std::_Deque_base< _Tp, _Alloc >::_M_initialize_map(), std::deque< _Tp, _Alloc >::_M_reallocate_map(), std::discard_block_engine< _RandomNumberEngine, __p, __r >::max(), std::shuffle_order_engine< _RandomNumberEngine, __k >::max(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

2.27.2.10 template<typename _Tp , typename _Compare > const _Tp & std::max (const _Tp & *__a,* const _Tp & *__b,* _Compare *__comp*) `[inline]`

This does what you think it does.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.
__comp	A comparison functor.

Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 260 of file stl_algobase.h.

2.27.2.11 template<typename _ForwardIterator > _ForwardIterator std::max_element (_ForwardIterator *__first,* _ForwardIterator *__last*)

Return the maximum element in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

Iterator referencing the first instance of the largest value.

Definition at line 6298 of file stl_algo.h.

2.27.2.12 template<typename _ForwardIterator , typename _Compare > _ForwardIterator std::max_element (_ForwardIterator *__first,* _ForwardIterator *__last,* _Compare *__comp*)

Return the maximum element in a range using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor.

Returns

Iterator referencing the first instance of the largest value according to __comp.

Definition at line 6326 of file stl_algo.h.

Referenced by std::valarray< _Tp >::max().

2.27.2.13 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > _OutputIterator std::merge (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result*)

Merges two sorted ranges.

Parameters

__first1	An iterator.
__first2	Another iterator.
__last1	Another iterator.
__last2	Another iterator.
__result	An iterator pointing to the end of the merged range.

Returns

An iterator pointing to the first element *not less than val*.

Merges the ranges [__first1,__last1) and [__first2,__last2) into the sorted range [__result, __result + (__last1-__first1) + (__last2-__first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 5540 of file stl_algo.h.

2.27.2.14 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > _OutputIterator std::merge (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result,* _Compare *__comp*)

Merges two sorted ranges.

Parameters

__first1	An iterator.
__first2	Another iterator.
__last1	Another iterator.
__last2	Another iterator.
__result	An iterator pointing to the end of the merged range.
__comp	A functor to use for comparisons.

Returns

An iterator pointing to the first element "not less than" *val*.

Merges the ranges [__first1,__last1) and [__first2,__last2) into the sorted range [__result, __result + (__last1-__first1) + (__last2-__first2)). Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5604 of file stl_algo.h.

2.27.2.15 template<typename _Tp > const _Tp & std::min (const _Tp & *__a,* const _Tp & *__b*) `[inline]`

This does what you think it does.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.

Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 193 of file stl_algobase.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort_qs_divide(), __gnu_profile::__report(), __gnu_parallel::__search_template(), __gnu_parallel::__sequential_random_shuffle(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare(), std::basic_string< char >::compare(), std::basic_string< _CharT, _Traits, _Alloc >::compare(), std::generate_canonical(), std::discard_block_engine< _RandomNumberEngine, __p, __r >::min(), std::shuffle_order_engine< _RandomNumberEngine, __k >::min(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), __gnu_cxx::random_sample_n(), std::basic_istream< _CharT, _Traits >::readsome(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind(), std::basic_string< _CharT, _Traits, _Alloc >::rfind(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_streambuf< _CharT, _Traits >::xsgetn(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and std::basic_streambuf< _CharT, _Traits >::xsputn().

2.27.2.16 template<typename _Tp , typename _Compare > const _Tp & std::min (const _Tp & *__a,* const _Tp & *__b,* _Compare *__comp*) [inline]

This does what you think it does.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.
__comp	A comparison functor.

Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 239 of file stl_algobase.h.

2.27.2.17 template<typename _ForwardIterator > _ForwardIterator std::min_element (_ForwardIterator *__first,* _ForwardIterator *__last*)

Return the minimum element in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

Iterator referencing the first instance of the smallest value.

Definition at line 6242 of file stl_algo.h.

2.27.2.18 template<typename _ForwardIterator , typename _Compare > _ForwardIterator std::min_element (_ForwardIterator *__first,* _ForwardIterator *__last,* _Compare *__comp*)

Return the minimum element in a range using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor.

Returns

Iterator referencing the first instance of the smallest value according to __comp.

Definition at line 6270 of file stl_algo.h.

Referenced by std::valarray< _Tp >::min().

2.27.2.19 template<typename _Tp > pair< const _Tp &, const _Tp & > std::minmax (const _Tp & *__a*, const _Tp & *__b*) [inline]

Determines min and max at once as an ordered pair.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.

Returns

A pair(__b, __a) if __b is smaller than __a, pair(__a, __b) otherwise.

Definition at line 4062 of file stl_algo.h.

2.27.2.20 template<typename _Tp , typename _Compare > pair< const _Tp &, const _Tp & > std::minmax (const _Tp & *__a*, const _Tp & *__b*, _Compare *__comp*) [inline]

Determines min and max at once as an ordered pair.

Parameters

__a	A thing of arbitrary type.
__b	Another thing of arbitrary type.
__comp	A comparison functor .

Returns

A pair(__b, __a) if __b is smaller than __a, pair(__a, __b) otherwise.

Definition at line 4082 of file stl_algo.h.

2.27.2.21 template<typename _ForwardIterator > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator *__first*, _ForwardIterator *__last*)

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

make_pair(m, M), where m is the first iterator i in [__first, __last) such that no other element in the range is smaller, and where M is the last iterator i in [__first, __last) such that no other element in the range is larger.

Definition at line 4101 of file stl_algo.h.

References std::make_pair().

2.27.2.22 template<typename _ForwardIterator , typename _Compare > pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator *__first,* _ForwardIterator *__last,* _Compare *__comp*)

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor.

Returns

> make_pair(m, M), where m is the first iterator i in [__first, __last) such that no other element in the range is smaller, and where M is the last iterator i in [__first, __last) such that no other element in the range is larger.

Definition at line 4177 of file stl_algo.h.

References std::make_pair().

2.27.2.23 template<typename _BidirectionalIterator > bool std::next_permutation (_BidirectionalIterator *__first,* _BidirectionalIterator *__last*)

Permute range into the next *dictionary* ordering.

Parameters

__first	Start of range.
__last	End of range.

Returns

> False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3677 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.27.2.24 template<typename _BidirectionalIterator , typename _Compare > bool std::next_permutation (_BidirectionalIterator *__first,* _BidirectionalIterator *__last,* _Compare *__comp*)

Permute range into the next *dictionary* ordering using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range [__first,__last) as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3734 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.27.2.25 template<typename _RandomAccessIterator > void std::nth_element (_RandomAccessIterator *__first,* _RandomAccessIterator *__nth,* _RandomAccessIterator *__last*) `[inline]`

Sort a sequence just enough to find a particular position.

Parameters

__first	An iterator.
__nth	Another iterator.
__last	Another iterator.

Returns

Nothing.

Rearranges the elements in the range [__first,__last) so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range [__first,__nth) and any iterator *j* in the range [__nth,__last) it holds that $*j < *i$ is false.

Definition at line 5384 of file stl_algo.h.

References std::__lg().

2.27.2.26 template<typename _RandomAccessIterator , typename _Compare > void std::nth_element (_RandomAccessIterator *__first,* _RandomAccessIterator *__nth,* _RandomAccessIterator *__last,* _Compare *__comp*) `[inline]`

Sort a sequence just enough to find a particular position using a predicate for comparison.

Parameters

__first	An iterator.
__nth	Another iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

Rearranges the elements in the range [__first,__last) so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range [__first,__nth) and any iterator *j* in the range [__nth,__last) it holds that `__comp(*j,*i)` is false.

Definition at line 5423 of file stl_algo.h.

References std::__lg().

2.27.2.27 template<typename _RandomAccessIterator > void std::partial_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__middle,* _RandomAccessIterator *__last*) [inline]

Sort the smallest elements of a sequence.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.

Returns

Nothing.

Sorts the smallest (__middle-__first) elements in the range [first,last) and moves them to the range [__first,__middle). The order of the remaining elements in the range [__middle,__last) is undefined. After the sort if *i* and *j* are iterators in the range [__first,__middle) such that i precedes j and *k* is an iterator in the range [__middle,__last) then ∗j<∗i and ∗k<∗i are both false.

Definition at line 5308 of file stl_algo.h.

References std::__heap_select(), and std::sort_heap().

2.27.2.28 template<typename _RandomAccessIterator , typename _Compare > void std::partial_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__middle,* _RandomAccessIterator *__last,* _Compare *__comp*) [inline]

Sort the smallest elements of a sequence using a predicate for comparison.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

Sorts the smallest (__middle-__first) elements in the range [__first,__last) and moves them to the range [__first,__middle). The order of the remaining elements in the range [__middle,__last) is undefined. After the sort if *i* and *j* are iterators in the range [__first,__middle) such that i precedes j and *k* is an iterator in the range [__middle,__last) then ∗__`comp`(j,∗i) and __`comp(*k,*i)` are both false.

Definition at line 5347 of file stl_algo.h.

References std::__heap_select(), and std::sort_heap().

2.27.2.29 template<typename _InputIterator , typename _RandomAccessIterator > _RandomAccessIterator std::partial_sort_copy (_InputIterator *__first,* _InputIterator *__last,* _RandomAccessIterator *__result_first,* _RandomAccessIterator *__result_last*)

Copy the smallest elements of a sequence.

Parameters

__first	An iterator.
__last	Another iterator.
__result_first	A random-access iterator.
__result_last	Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range [__first,__last) to the range beginning at `___result_first`, where the number of elements to be copied, `N`, is the smaller of (__last-__first) and (__result_last-__result_first). After the sort if *i* and *j* are iterators in the range [__result_first,__result_first+N) such that i precedes j then ∗j<∗i is false. The value returned is `___result_first+N`.

Definition at line 1998 of file stl_algo.h.

References std::make_heap(), and std::sort_heap().

2.27.2.30 template<typename _InputIterator , typename _RandomAccessIterator , typename _Compare > _RandomAccessIterator std::partial_sort_copy (_InputIterator *__first,* _InputIterator *__last,* _RandomAccessIterator *__result_first,* _RandomAccessIterator *__result_last,* _Compare *__comp*)

Copy the smallest elements of a sequence using a predicate for comparison.

Parameters

__first	An input iterator.
__last	Another input iterator.
__result_first	A random-access iterator.
__result_last	Another random-access iterator.
__comp	A comparison functor.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range [__first,__last) to the range beginning at `result_first`, where the number of elements to be copied, `N`, is the smaller of (__last-__first) and (__result_last-__result_first). After the sort if *i* and *j* are iterators in the range [__result_first,__result_first+N) such that i precedes j then `___comp(*j,*i)` is false. The value returned is `___result_first+N`.

Definition at line 2064 of file stl_algo.h.

References std::make_heap(), and std::sort_heap().

2.27.2.31 template<typename _BidirectionalIterator > bool std::prev_permutation (_BidirectionalIterator *__first,* _BidirectionalIterator *__last*)

Permute range into the previous *dictionary* ordering.

Parameters

__first	Start of range.
__last	End of range.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3790 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.27.2.32 template<typename _BidirectionalIterator , typename _Compare > bool std::prev_permutation (_BidirectionalIterator *__first,* _BidirectionalIterator *__last,* _Compare *__comp*)

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range [__first,__last) as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3847 of file stl_algo.h.

References std::iter_swap(), and std::reverse().

2.27.2.33 template<typename _RandomAccessIterator > void std::sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) [inline]

Sort the elements of a sequence.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

Nothing.

Sorts the elements in the range [__first,__last) in ascending order, such that for each iterator *i* in the range [__first,__last-1), ∗(i+1)<∗i is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5461 of file stl_algo.h.

References std::__final_insertion_sort(), std::__introsort_loop(), and std::__lg().

2.27.2.34 template<typename _RandomAccessIterator , typename _Compare > void std::sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*) `[inline]`

Sort the elements of a sequence using a predicate for comparison.

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

Sorts the elements in the range [__first,__last) in ascending order, such that `__comp`(∗(i+1),∗i) is false for every iterator *i* in the range [__first,__last-1).

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 5497 of file stl_algo.h.

References std::__final_insertion_sort(), std::__introsort_loop(), and std::__lg().

2.27.2.35 template<typename _RandomAccessIterator > void std::stable_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) `[inline]`

Sort the elements of a sequence, preserving the relative order of equivalent elements.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

Nothing.

Sorts the elements in the range [__first,__last) in ascending order, such that for each iterator `i` in the range [__first,__last-1), ∗(i+1)<∗i is false.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range [__first,__last) such that `x<y` is false and `y<x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5663 of file stl_algo.h.

References std::__inplace_stable_sort(), std::_Temporary_buffer< _ForwardIterator, _Tp >::begin(), and std::_Temporary_buffer< _ForwardIterator, _Tp >::size().

2.27.2.36 template<typename _RandomAccessIterator , typename _Compare > void std::stable_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*) `[inline]`

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

Sorts the elements in the range [__first,__last) in ascending order, such that for each iterator `i` in the range [__first,__last-1), ___`comp`(∗(i+1),∗i) is false.

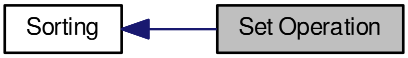
The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range [__first,__last) such that ___`comp(x,y)` is false and ___`comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5705 of file stl_algo.h.

References std::__inplace_stable_sort(), std::_Temporary_buffer< _ForwardIterator, _Tp >::begin(), and std::_Temporary_buffer< _ForwardIterator, _Tp >::size().

2.28 Set Operation

Collaboration diagram for Set Operation:



Functions

- template<typename _InputIterator1 , typename _InputIterator2 >
  bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare >
  bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)

2.28.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

2.28.2 Function Documentation

2.28.2.1 template<typename _InputIterator1 , typename _InputIterator2 > bool std::includes (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2*)

Determines whether all elements of a sequence exists in a range.

Parameters

__first1	Start of search range.
__last1	End of search range.
__first2	Start of sequence
__last2	End of sequence.

Returns

True if each element in [__first2,__last2) is contained in order within [__first1,__last1). False otherwise.

This operation expects both [__first1,__last1) and [__first2,__last2) to be sorted. Searches for the presence of each element in [__first2,__last2) within [__first1,__last1). The iterators over each range only move forward, so this is a linear algorithm. If an element in [__first2,__last2) is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 3572 of file stl_algo.h.

2.28.2.2 template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare > bool std::includes (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _Compare *__comp*)

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

__first1	Start of search range.
__last1	End of search range.
__first2	Start of sequence
__last2	End of sequence.
__comp	Comparison function to use.

Returns

True if each element in [__first2,__last2) is contained in order within [__first1,__last1) according to comp. False otherwise.

This operation expects both [__first1,__last1) and [__first2,__last2) to be sorted. Searches for the presence of each element in [__first2,__last2) within [__first1,__last1), using comp to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [__first2,__last2) is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 3623 of file stl_algo.h.

2.28.2.3 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > _OutputIterator std::set_difference (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result*)

Return the difference of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6001 of file stl_algo.h.

2.28.2.4 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > _OutputIterator std::set_difference (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result,* _Compare *__comp*)

Return the difference of two sorted ranges using comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6062 of file stl_algo.h.

2.28.2.5 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > _OutputIterator std::set_intersection (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result*)

Return the intersection of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5886 of file stl_algo.h.

2.28.2.6 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > _OutputIterator std::set_intersection (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result,* _Compare *__comp*)

Return the intersection of two sorted ranges using comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5943 of file stl_algo.h.

2.28.2.7 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > _OutputIterator std::set_symmetric_difference (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result*)

Return the symmetric difference of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6120 of file stl_algo.h.

2.28.2.8 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > _OutputIterator std::set_symmetric_difference (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result,* _Compare *__comp*)

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 6186 of file stl_algo.h.

2.28.2.9 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > _OutputIterator std::set_union (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result*)

Return the union of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5752 of file stl_algo.h.

2.28.2.10 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > _OutputIterator std::set_union (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result,* _Compare *__comp*)

Return the union of two sorted ranges using a comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__comp	The comparison functor.

Returns

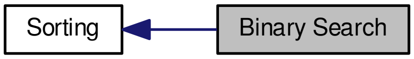
End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5819 of file stl_algo.h.

2.29 Binary Search

Collaboration diagram for Binary Search:



Functions

- template<typename _ForwardIterator , typename _Tp >
  bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _ForwardIterator , typename _Tp >
  pair< _ForwardIterator,
  _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  pair< _ForwardIterator,
  _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)

2.29.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

2.29.2 Function Documentation

2.29.2.1 template<typename _ForwardIterator , typename _Tp > bool std::binary_search (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__val*)

Determines whether an element exists in a range.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

True if `__val` (or its equivalent) is in [`__first`,`__last`].

Note that this does not actually return an iterator to `__val`. For that, use std::find or a container's specialized find member functions.

Definition at line 2720 of file stl_algo.h.

References std::lower_bound().

2.29.2.2 template<typename _ForwardIterator , typename _Tp , typename _Compare > bool std::binary_search (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__val,* _Compare *__comp*)

Determines whether an element exists in a range.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.
__comp	A functor to use for comparisons.

Returns

True if `__val` (or its equivalent) is in [__first,__last].

Note that this does not actually return an iterator to `__val`. For that, use std::find or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2753 of file stl_algo.h.

References std::lower_bound().

2.29.2.3 template<typename _ForwardIterator , typename _Tp > pair<_ForwardIterator, _ForwardIterator> std::equal_range (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__val*)

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

Definition at line 2597 of file stl_algo.h.

References std::advance(), std::distance(), std::lower_bound(), and std::upper_bound().

2.29.2.4 template<typename _ForwardIterator , typename _Tp , typename _Compare > pair<_ForwardIterator, _ForwardIterator> std::equal_range (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__val,* _Compare *__comp*)

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.
__comp	A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

Definition at line 2659 of file stl_algo.h.

References std::advance(), std::distance(), std::lower_bound(), and std::upper_bound().

2.29.2.5 template<typename _ForwardIterator , typename _Tp > _ForwardIterator std::lower_bound (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__val*)

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

An iterator pointing to the first element *not less than val*, or end() if every element is less than *val*.

Definition at line 943 of file stl_algobase.h.

References std::advance(), and std::distance().

Referenced by std::__merge_adaptive(), std::__merge_without_buffer(), std::binary_search(), and std::equal_range().

2.29.2.6 template<typename _ForwardIterator , typename _Tp , typename _Compare > _ForwardIterator std::lower_bound (_ForwardIterator *__first*, _ForwardIterator *__last*, const _Tp & *__val*, _Compare *__comp*)

Finds the first position in which `__val` could be inserted without changing the ordering.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.
__comp	A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than* `__val`, or end() if every element is less than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2448 of file stl_algo.h.

References std::advance(), and std::distance().

2.29.2.7 template<typename _ForwardIterator , typename _Tp > _ForwardIterator std::upper_bound (_ForwardIterator *__first*, _ForwardIterator *__last*, const _Tp & *__val*)

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

An iterator pointing to the first element greater than `__val`, or end() if no elements are greater than `__val`.

Definition at line 2495 of file stl_algo.h.

References std::advance(), and std::distance().

2.29.2.8 template<typename _ForwardIterator , typename _Tp , typename _Compare > _ForwardIterator std::upper_bound (_ForwardIterator *__first*, _ForwardIterator *__last*, const _Tp & *__val*, _Compare *__comp*)

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.
__comp	A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than `__val`, or end() if no elements are greater than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2544 of file stl_algo.h.

References std::advance(), and std::distance().

Referenced by std::__merge_adaptive(), std::__merge_without_buffer(), and std::equal_range().

2.30 Allocators

Collaboration diagram for Allocators:

Classes

- struct __gnu_cxx::__alloc_traits< _Alloc >

  Uniform interface to C++98 and C++0x allocators.
- class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

  This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html`.
- class __gnu_cxx::__pool_alloc< _Tp >

  Allocator using a memory pool with a single lock.
- class __gnu_cxx::_ExtPtr_allocator< _Tp >

  An example allocator which uses a non-standard pointer type.
  This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.
- class __gnu_cxx::array_allocator< _Tp, _Array >

  An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class __gnu_cxx::bitmap_allocator< _Tp >

  Bitmap Allocator, primary template.
- class __gnu_cxx::debug_allocator< _Alloc >

  A meta-allocator with debugging bits, as per [20.4].
  This is precisely the allocator defined in the C++ Standard.
- class __gnu_cxx::malloc_allocator< _Tp >

  An allocator that uses malloc.
  This is precisely the allocator defined in the C++ Standard.
- class __gnu_cxx::new_allocator< _Tp >

  An allocator that uses global new, as per [20.4].
  This is precisely the allocator defined in the C++ Standard.
- class __gnu_cxx::throw_allocator_base< _Tp, _Cond >

  Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templatized code.
  Note: Deallocate not allowed to throw.
- class std::allocator< void >

  allocator<void> specialization.
- struct std::allocator_traits< _Alloc >

  Uniform interface to all allocator types.
- struct uses_allocator< typename, typename >

  Declare uses_allocator so it can be specialized in <queue> etc.

Typedefs

- template<typename _Tp >
  using std::__allocator_base = __gnu_cxx::new_allocator< _Tp >

Functions

- template<typename _Alloc >
  auto **std::__do_outermost** (_Alloc &__a, _Alloc ∗) -> decltype(__a.outer_allocator())
- template<typename _Alloc >
  _Alloc & **std::__do_outermost** (_Alloc &__a,...)
- template<typename _Alloc >
  auto **std::__outermost** (_Alloc &__a) -> decltype(__do_outermost(__a,&__a))
- template<typename _T1 , typename _T2 >
  bool **std::operator!=** (const allocator< _T1 > &, const allocator< _T2 > &)
- template<typename _Tp >
  bool **std::operator!=** (const allocator< _Tp > &, const allocator< _Tp > &)
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **std::operator!=** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept
- template<typename _T1 , typename _T2 >
  bool **std::operator==** (const allocator< _T1 > &, const allocator< _T2 > &)
- template<typename _Tp >
  bool **std::operator==** (const allocator< _Tp > &, const allocator< _Tp > &)
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **std::operator==** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept

2.30.1 Detailed Description

Classes encapsulating memory operations.

2.30.2 Typedef Documentation

2.30.2.1 template<typename _Tp > using std::__allocator_base = typedef __gnu_cxx::new_allocator<_Tp>

An alias to the base class for std::allocator.

Used to set the std::allocator base class to __gnu_cxx::new_allocator.

Template Parameters

_Tp	Type of allocated object.

Definition at line 48 of file c++allocator.h.

2.31 Atomics

Classes

- struct std::__atomic_base< _PTp $*$ >

  Partial specialization for pointer types.
- struct std::__atomic_flag_base

  Base type for atomic_flag.
- struct std::atomic< _Tp >

  Generic atomic type, primary class template.
- struct std::atomic< _Tp $*$ >

  Partial specialization for pointer types.
- struct std::atomic< bool >

  Explicit specialization for bool.
- struct std::atomic< char >

  Explicit specialization for char.
- struct std::atomic< char16_t >

  Explicit specialization for char16_t.
- struct std::atomic< char32_t >

  Explicit specialization for char32_t.
- struct std::atomic< int >

  Explicit specialization for int.
- struct std::atomic< long >

  Explicit specialization for long.
- struct std::atomic< long long >

  Explicit specialization for long long.
- struct std::atomic< short >

  Explicit specialization for short.
- struct std::atomic< signed char >

  Explicit specialization for signed char.
- struct std::atomic< unsigned char >

  Explicit specialization for unsigned char.
- struct std::atomic< unsigned int >

  Explicit specialization for unsigned int.
- struct std::atomic< unsigned long >

  Explicit specialization for unsigned long.
- struct std::atomic< unsigned long long >

  Explicit specialization for unsigned long long.
- struct std::atomic< unsigned short >

  Explicit specialization for unsigned short.
- struct std::atomic< wchar_t >

  Explicit specialization for wchar_t.
- struct std::atomic_bool

  atomic_bool
- struct std::atomic_flag

  atomic_flag

Macros

- #define ATOMIC_BOOL_LOCK_FREE
- #define **ATOMIC_CHAR16_T_LOCK_FREE**
- #define **ATOMIC_CHAR32_T_LOCK_FREE**
- #define **ATOMIC_CHAR_LOCK_FREE**
- #define **ATOMIC_FLAG_INIT**
- #define **ATOMIC_INT_LOCK_FREE**
- #define **ATOMIC_LLONG_LOCK_FREE**
- #define **ATOMIC_LONG_LOCK_FREE**
- #define **ATOMIC_POINTER_LOCK_FREE**
- #define **ATOMIC_SHORT_LOCK_FREE**
- #define **ATOMIC_VAR_INIT**(_VI)
- #define **ATOMIC_WCHAR_T_LOCK_FREE**

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**
- typedef __atomic_base< char > std::atomic_char
- typedef __atomic_base< char16_t > std::atomic_char16_t
- typedef __atomic_base< char32_t > std::atomic_char32_t
- typedef __atomic_base< int > std::atomic_int
- typedef __atomic_base < int_fast16_t > std::atomic_int_fast16_t
- typedef __atomic_base < int_fast32_t > std::atomic_int_fast32_t
- typedef __atomic_base < int_fast64_t > std::atomic_int_fast64_t
- typedef __atomic_base < int_fast8_t > std::atomic_int_fast8_t
- typedef __atomic_base < int_least16_t > std::atomic_int_least16_t
- typedef __atomic_base < int_least32_t > std::atomic_int_least32_t
- typedef __atomic_base < int_least64_t > std::atomic_int_least64_t
- typedef __atomic_base < int_least8_t > std::atomic_int_least8_t
- typedef __atomic_base< intmax_t > std::atomic_intmax_t
- typedef __atomic_base< intptr_t > std::atomic_intptr_t
- typedef __atomic_base< long long > std::atomic_llong
- typedef __atomic_base< long > std::atomic_long
- typedef __atomic_base< ptrdiff_t > std::atomic_ptrdiff_t
- typedef __atomic_base< signed char > std::atomic_schar
- typedef __atomic_base< short > std::atomic_short
- typedef __atomic_base< size_t > std::atomic_size_t
- typedef __atomic_base < unsigned char > std::atomic_uchar
- typedef __atomic_base < unsigned int > std::atomic_uint

- typedef __atomic_base
  < uint_fast16_t > std::atomic_uint_fast16_t
- typedef __atomic_base
  < uint_fast32_t > std::atomic_uint_fast32_t
- typedef __atomic_base
  < uint_fast64_t > std::atomic_uint_fast64_t
- typedef __atomic_base
  < uint_fast8_t > std::atomic_uint_fast8_t
- typedef __atomic_base
  < uint_least16_t > std::atomic_uint_least16_t
- typedef __atomic_base
  < uint_least32_t > std::atomic_uint_least32_t
- typedef __atomic_base
  < uint_least64_t > std::atomic_uint_least64_t
- typedef __atomic_base
  < uint_least8_t > std::atomic_uint_least8_t
- typedef __atomic_base< uintmax_t > std::atomic_uintmax_t
- typedef __atomic_base< uintptr_t > std::atomic_uintptr_t
- typedef __atomic_base
  < unsigned long long > std::atomic_ullong
- typedef __atomic_base
  < unsigned long > std::atomic_ulong
- typedef __atomic_base
  < unsigned short > std::atomic_ushort
- typedef __atomic_base< wchar_t > std::atomic_wchar_t
- typedef enum std::memory_order std::memory_order

Enumerations

- enum **__memory_order_modifier** { **__memory_order_mask**, **__memory_order_modifier_mask**, **__memory_order_hle_acquire**, **__memory_order_hle_release** }
- enum std::memory_order {
  memory_order_relaxed, **memory_order_consume**, **memory_order_acquire**, **memory_order_release**, **memory_order_acq_rel**, **memory_order_seq_cst** }

Functions

- constexpr memory_order **std::__cmpexch_failure_order** (memory_order __m) noexcept
- constexpr memory_order **std::__cmpexch_failure_order2** (memory_order __m) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong_explicit** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept

- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak_explicit** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange_explicit** (atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange_explicit** (volatile atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add_explicit** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add_explicit** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept

- template<typename _ITp >
  _ITp **std::atomic_fetch_sub** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub_explicit** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub_explicit** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- void **std::atomic_flag_clear** (atomic_flag ∗__a) noexcept
- void **std::atomic_flag_clear** (volatile atomic_flag ∗__a) noexcept
- void **std::atomic_flag_clear_explicit** (atomic_flag ∗__a, memory_order __m) noexcept
- void **std::atomic_flag_clear_explicit** (volatile atomic_flag ∗__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set** (atomic_flag ∗__a) noexcept
- bool **std::atomic_flag_test_and_set** (volatile atomic_flag ∗__a) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (atomic_flag ∗__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (volatile atomic_flag ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  void **std::atomic_init** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **std::atomic_init** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  bool **std::atomic_is_lock_free** (const atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  bool **std::atomic_is_lock_free** (const volatile atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load** (const atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load** (const volatile atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load_explicit** (const atomic< _ITp > ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load_explicit** (const volatile atomic< _ITp > ∗__a, memory_order __m) noexcept
- void **std::atomic_signal_fence** (memory_order __m) noexcept

- template<typename _ITp >
  void **std::atomic_store** (atomic< _ITp > *__a, _ITp __i) noexcept
- template<typename _ITp >
  void **std::atomic_store** (volatile atomic< _ITp > *__a, _ITp __i) noexcept
- template<typename _ITp >
  void **std::atomic_store_explicit** (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  void **std::atomic_store_explicit** (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept
- void **std::atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Tp >
  _Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order **std::operator&** (memory_order __m, __memory_order_modifier __mod)
- constexpr memory_order **std::operator|** (memory_order __m, __memory_order_modifier __mod)

2.31.1 Detailed Description

Components for performing atomic operations.

2.31.2 Macro Definition Documentation

2.31.2.1 #define ATOMIC_BOOL_LOCK_FREE

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

Definition at line 49 of file atomic_lockfree_defines.h.

2.31.3 Typedef Documentation

2.31.3.1 typedef __atomic_base<char> std::atomic_char

atomic_char

Definition at line 117 of file atomic_base.h.

2.31.3.2 typedef __atomic_base<char16_t> std::atomic_char16_t

atomic_char16_t

Definition at line 156 of file atomic_base.h.

2.31.3.3 typedef __atomic_base< char32_t > std::atomic_char32_t

atomic_char32_t

Definition at line 159 of file atomic_base.h.

2.31.3.4 typedef __atomic_base<int> std::atomic_int

atomic_int

Definition at line 135 of file atomic_base.h.

2.31.3.5 typedef __atomic_base<int_fast16_t> std::atomic_int_fast16_t

atomic_int_fast16_t

Definition at line 197 of file atomic_base.h.

2.31.3.6 typedef __atomic_base<int_fast32_t> std::atomic_int_fast32_t

atomic_int_fast32_t

Definition at line 203 of file atomic_base.h.

2.31.3.7 typedef __atomic_base<int_fast64_t> std::atomic_int_fast64_t

atomic_int_fast64_t

Definition at line 209 of file atomic_base.h.

2.31.3.8 typedef __atomic_base<int_fast8_t> std::atomic_int_fast8_t

atomic_int_fast8_t

Definition at line 191 of file atomic_base.h.

2.31.3.9 typedef __atomic_base<int_least16_t> std::atomic_int_least16_t

atomic_int_least16_t

Definition at line 172 of file atomic_base.h.

2.31.3.10 typedef __atomic_base<int_least32_t> std::atomic_int_least32_t

atomic_int_least32_t

Definition at line 178 of file atomic_base.h.

2.31.3.11 typedef __atomic_base<int_least64_t> std::atomic_int_least64_t

atomic_int_least64_t

Definition at line 184 of file atomic_base.h.

2.31.3.12 typedef __atomic_base<int_least8_t> std::atomic_int_least8_t

atomic_int_least8_t

Definition at line 166 of file atomic_base.h.

2.31.3.13 typedef __atomic_base<intmax_t> std::atomic_intmax_t

atomic_intmax_t

Definition at line 225 of file atomic_base.h.

2.31.3.14 typedef __atomic_base<intptr_t> std::atomic_intptr_t

atomic_intptr_t

Definition at line 216 of file atomic_base.h.

2.31.3.15 typedef __atomic_base<long long> std::atomic_llong

atomic_llong

Definition at line 147 of file atomic_base.h.

2.31.3.16 typedef __atomic_base<long> std::atomic_long

atomic_long

Definition at line 141 of file atomic_base.h.

2.31.3.17 typedef __atomic_base<ptrdiff_t> std::atomic_ptrdiff_t

atomic_ptrdiff_t

Definition at line 231 of file atomic_base.h.

2.31.3.18 typedef __atomic_base<signed char> std::atomic_schar

atomic_schar

Definition at line 123 of file atomic_base.h.

2.31.3.19 typedef __atomic_base<short> std::atomic_short

atomic_short

Definition at line 129 of file atomic_base.h.

2.31.3.20 typedef __atomic_base<size_t> std::atomic_size_t

atomic_size_t

Definition at line 222 of file atomic_base.h.

2.31.3.21 typedef __atomic_base<unsigned char> std::atomic_uchar

atomic_uchar

Definition at line 126 of file atomic_base.h.

2.31.3.22 typedef __atomic_base<unsigned int> std::atomic_uint

atomic_uint

Definition at line 138 of file atomic_base.h.

2.31.3.23 typedef __atomic_base<uint_fast16_t> std::atomic_uint_fast16_t

atomic_uint_fast16_t

Definition at line 200 of file atomic_base.h.

2.31.3.24 typedef __atomic_base<uint_fast32_t> std::atomic_uint_fast32_t

atomic_uint_fast32_t

Definition at line 206 of file atomic_base.h.

2.31.3.25 typedef __atomic_base< uint_fast64_t > std::atomic_uint_fast64_t

atomic_uint_fast64_t

Definition at line 212 of file atomic_base.h.

2.31.3.26 typedef __atomic_base< uint_fast8_t > std::atomic_uint_fast8_t

atomic_uint_fast8_t

Definition at line 194 of file atomic_base.h.

2.31.3.27 typedef __atomic_base< uint_least16_t > std::atomic_uint_least16_t

atomic_uint_least16_t

Definition at line 175 of file atomic_base.h.

2.31.3.28 typedef __atomic_base< uint_least32_t > std::atomic_uint_least32_t

atomic_uint_least32_t

Definition at line 181 of file atomic_base.h.

2.31.3.29 typedef __atomic_base< uint_least64_t > std::atomic_uint_least64_t

atomic_uint_least64_t

Definition at line 187 of file atomic_base.h.

2.31.3.30 typedef __atomic_base< uint_least8_t > std::atomic_uint_least8_t

atomic_uint_least8_t

Definition at line 169 of file atomic_base.h.

2.31.3.31 typedef __atomic_base< uintmax_t > std::atomic_uintmax_t

atomic_uintmax_t

Definition at line 228 of file atomic_base.h.

2.31.3.32 typedef __atomic_base< uintptr_t > std::atomic_uintptr_t

atomic_uintptr_t

Definition at line 219 of file atomic_base.h.

2.31.3.33 typedef __atomic_base< unsigned long long > std::atomic_ullong

atomic_ullong

Definition at line 150 of file atomic_base.h.

2.31.3.34 typedef __atomic_base< unsigned long > std::atomic_ulong

atomic_ulong

Definition at line 144 of file atomic_base.h.

2.31.3.35 typedef __atomic_base<unsigned short> std::atomic_ushort

atomic_ushort

Definition at line 132 of file atomic_base.h.

2.31.3.36 typedef __atomic_base<wchar_t> std::atomic_wchar_t

atomic_wchar_t

Definition at line 153 of file atomic_base.h.

2.31.3.37 typedef enum std::memory_order std::memory_order

Enumeration for memory_order.

2.31.4 Enumeration Type Documentation

2.31.4.1 enum std::memory_order

Enumeration for memory_order.

Definition at line 52 of file atomic_base.h.

2.31.5 Function Documentation

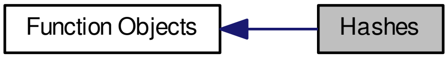
2.31.5.1 template<typename _Tp > _Tp std::kill_dependency (_Tp __*y*) `[inline]`, `[noexcept]`

kill_dependency

Definition at line 108 of file atomic_base.h.

2.32 Hashes

Collaboration diagram for Hashes:



Classes

- struct hash< _Tp >

  Primary class template hash.
- struct std::hash< _Tp * >

  Partial specializations for pointer types.
- struct std::hash< bool >

  Explicit specialization for bool.
- struct std::hash< char >

  Explicit specialization for char.
- struct std::hash< char16_t >

  Explicit specialization for char16_t.
- struct std::hash< char32_t >

  Explicit specialization for char32_t.
- struct std::hash< double >

  Specialization for double.
- struct std::hash< float >

  Specialization for float.
- struct std::hash< int >

  Explicit specialization for int.
- struct std::hash< long >

  Explicit specialization for long.
- struct std::hash< long double >

  Specialization for long double.
- struct std::hash< long long >

  Explicit specialization for long long.
- struct std::hash< short >

  Explicit specialization for short.
- struct std::hash< signed char >

  Explicit specialization for signed char.
- struct std::hash< unsigned char >

  Explicit specialization for unsigned char.
- struct std::hash< unsigned int >

  Explicit specialization for unsigned int.

- struct std::hash< unsigned long >

  Explicit specialization for unsigned long.
- struct std::hash< unsigned long long >

  Explicit specialization for unsigned long long.
- struct std::hash< unsigned short >

  Explicit specialization for unsigned short.
- struct std::hash< wchar_t >

  Explicit specialization for wchar_t.

Macros

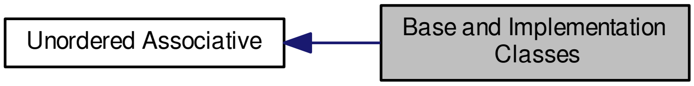
- #define **_Cxx_hashtable_define_trivial_hash**(_Tp)

2.32.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

2.33 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
- struct _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
- struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct _Hash_node< _Value, _Cache_hash_code >
- struct _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >
- struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys >
- struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct _Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- struct std::__detail::_Before_begin< _NodeAlloc >
- struct std::__detail::_Default_ranged_hash

  Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that h(k, N) = h2(h1(k), N), but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.
- struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >

  Specialization.
- struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >

  Specialization.
- struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

  Specialization.
- struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

  Specialization.
- struct std::__detail::_Equality_base
- struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >

  Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.
- struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >

  Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.
- struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

- struct std::__detail::_Hash_node< _Value, false >
- struct std::__detail::_Hash_node< _Value, true >
- struct std::__detail::_Hash_node_base
- struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >

  Specialization not using EBO.
- struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >

  Specialization using EBO.
- struct std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
- struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >

  Specialization.
- struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >

  Specialization.
- struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >

  Specialization.
- struct std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- struct std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

  local const_iterators
- struct std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

  local iterators
- struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

  Specialization.
- struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >

  Specialization.
- struct std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
- struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

  Partial specialization, __unique_keys set to false.
- struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

  Partial specialization, __unique_keys set to true.
- struct std::__detail::_Mod_range_hashing

  Default range hashing function: use division to fold a large number into the range [0, N).
- struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >

  Node const_iterators, used to iterate through all the hashtable.
- struct std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >

  Node iterators, used to iterate through all the hashtable.
- struct std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >

  Base class for node iterators.
- struct std::__detail::_Prime_rehash_policy

  Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

- struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >

  Specialization.
- class std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Functions

- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **std::__detail::__distance_fw** (_Iterator __first, _Iterator __last, std::input_iterator_tag)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **std::__detail::__distance_fw** (_Iterator __first, _Iterator __last, std::forward_iterator_tag)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **std::__detail::__distance_fw** (_Iterator __first, _Iterator __last)
- bool **std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::_M_equal** (const __hashtable &) const
- bool **std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >::_M_equal** (const __hashtable &) const
- template<typename _Uiterator >
  static bool **std::__detail::_Equality_base::_S_is_permutation** (_Uiterator, _Uiterator, _Uiterator)
- mapped_type & **std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::at** (const key_type &__k)
- const mapped_type & **std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::at** (const key_type &__k) const
- template<typename _InputIterator >
  void **std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Value , bool _Cache_hash_code>
  bool **std::__detail::operator!=** (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)
- template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>
  bool **std::__detail::operator!=** (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)
- template<typename _Value , bool _Cache_hash_code>
  bool **std::__detail::operator==** (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)
- template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>
  bool **std::__detail::operator==** (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)
- mapped_type & **std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::operator[]** (const key_type &__k)
- mapped_type & **std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::operator[]** (key_type &&__k)

2.33.1 Detailed Description

2.34 Locales

Classes

- class std::codecvt< _InternT, _ExternT, _StateT >

  Primary class template codecvt.
  NB: Generic, mostly useless implementation.
- class std::ctype< _CharT >

  Primary class template ctype facet.
  This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.
- class std::ctype< char >

  The ctype<char> specialization.
  This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.
- class std::ctype< wchar_t >

  The ctype<wchar_t> specialization.
  This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
- class std::locale

  Container class for localization functionality.
  The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.
- class std::locale::facet

  Localization functionality base class.
  The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.
- class std::locale::id

  Facet ID class.
  The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.
- class std::messages< _CharT >

  Primary class template messages.
  This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.
- struct std::messages_base

  Messages facet base class providing catalog typedef.
- class std::money_base

  Money format ordering data.
  This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.
- class std::money_get< _CharT, _InIter >

  Primary class template money_get.
  This facet encapsulates the code to parse and return a monetary amount from a string.
- class std::money_put< _CharT, _OutIter >

  Primary class template money_put.
  This facet encapsulates the code to format and output a monetary amount.
- class std::moneypunct< _CharT, _Intl >

  Primary class template moneypunct.
  This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class std::num_get< _CharT, _InIter >

  Primary class template num_get.
  This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
- class std::num_put< _CharT, _OutIter >

  Primary class template num_put.
  This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.
- class std::numpunct< _CharT >

  Primary class template numpunct.
  This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
- class std::time_base

  Time format ordering data.
  This class provides an enum representing different orderings of time: day, month, and year.
- class std::time_get< _CharT, _InIter >

  Primary class template time_get.
  This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.
- class std::time_put< _CharT, _OutIter >

  Primary class template time_put.
  This facet encapsulates the code to format and output dates and times according to formats used by strftime().

Functions

- template<typename _Facet >
  bool std::has_facet (const locale &__loc) throw ()
- template<typename _Facet >
  const _Facet & std::use_facet (const locale &__loc)

2.34.1 Detailed Description

Classes and functions for internationalization and localization.

2.34.2 Function Documentation

2.34.2.1 template<typename _Facet > bool std::has_facet (const locale & *__loc*) throw ()

Test for the presence of a facet.

has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if `__loc` contains a facet of type _Facet, else false.

Definition at line 104 of file locale_classes.tcc.

2.34.2.2 template<typename _Facet > const _Facet & std::use_facet (const locale & *__loc*)

Return a facet.

use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

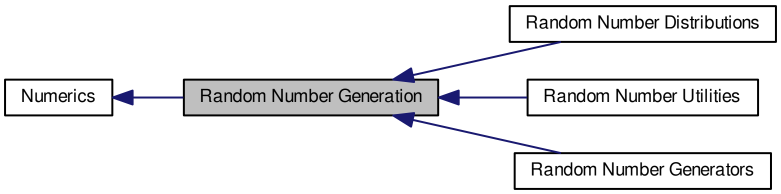
Exceptions

std::bad_cast	if `__loc` doesn't contain a facet of type _Facet.

Definition at line 132 of file locale_classes.tcc.

2.35 Random Number Generation

Collaboration diagram for Random Number Generation:



Modules

- Random Number Distributions
- Random Number Generators
- Random Number Utilities

Namespaces

- namespace std::__detail

Functions

- template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator >
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)

2.35.1 Detailed Description

A facility for generating random numbers on selected distributions.

2.35.2 Function Documentation

2.35.2.1 template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator & *__g*)

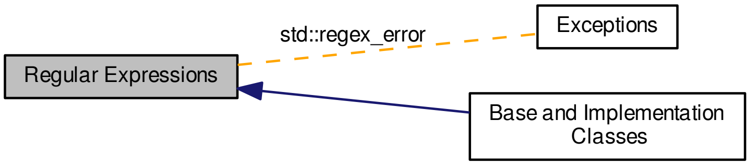
A function template for converting the output of a (integral) uniform random number generator to a floatng point result in the range [0-1).

Definition at line 3462 of file bits/random.tcc.

References std::log(), and std::min().

2.36 Regular Expressions

Collaboration diagram for Regular Expressions:



Modules

- Base and Implementation Classes

Namespaces

- namespace std::regex_constants

Classes

- class std::basic_regex< _Ch_type, _Rx_traits >
- class std::match_results< _Bi_iter, _Alloc >

  The results of a match or search operation.
- class std::regex_error

  A regular expression exception class.
  The regular expression library throws objects of this class on error.
- class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- struct std::regex_traits< _Ch_type >

  Class regex_traits. Describes aspects of a regular expression.
- class std::sub_match< _BiIter >

Typedefs

- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  using **std::__sub_match_string** = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >
- typedef match_results< const
  char ∗ > **std::cmatch**
- typedef regex_iterator< const
  char ∗ > **std::cregex_iterator**
- typedef regex_token_iterator
  < const char ∗ > std::cregex_token_iterator

- typedef sub_match< const char ∗ > std::csub_match
- typedef basic_regex< char > std::regex
- typedef match_results
  < string::const_iterator > **std::smatch**
- typedef regex_iterator
  < string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator
  < string::const_iterator > std::sregex_token_iterator
- typedef sub_match
  < string::const_iterator > std::ssub_match
- typedef match_results< const
  wchar_t ∗ > **std::wcmatch**
- typedef regex_iterator< const
  wchar_t ∗ > **std::wcregex_iterator**
- typedef regex_token_iterator
  < const wchar_t ∗ > std::wcregex_token_iterator
- typedef sub_match< const
  wchar_t ∗ > std::wcsub_match
- typedef basic_regex< wchar_t > std::wregex
- typedef match_results
  < wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator
  < wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator
  < wstring::const_iterator > std::wsregex_token_iterator
- typedef sub_match
  < wstring::const_iterator > std::wssub_match

Functions

- template<typename _Bi_iter >
  const sub_match< _Bi_iter > & **std::__unmatched_sub** ()
- bool std::regex_traits< _Ch_type >::isctype (_Ch_type __c, char_class_type __f) const
- template<typename _BiIter >
  bool std::operator!= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)

- template<typename _Bi_iter , class _Alloc >
  bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _BiIter >
  bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)
- template<typename _Bi_iter >
  bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter >
  basic_ostream< _Ch_type,
  _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)
- template<typename _BiIter >
  bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _BiIter >
  bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)

- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Bi_iter , typename _Alloc >
  bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _BiIter >
  bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _BiIter >
  bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)

- template<typename _Bi_iter >
  bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Ch_type , typename _Rx_traits >
  void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)
- template<typename _Bi_iter , typename _Alloc >
  void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)
- int std::regex_traits< _Ch_type >::value (_Ch_type __ch, int __radix) const

Matching, Searching, and Replacing

- template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , typename _Alloc , typename _Rx_traits >
  bool std::regex_match (const _Ch_type ∗__s, match_results< const _Ch_type ∗, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , class _Rx_traits >
  bool std::regex_match (const _Ch_type ∗__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _Str_allocator , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , class _Alloc , class _Rx_traits >
  bool std::regex_search (const _Ch_type ∗__s, match_results< const _Ch_type ∗, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (const _Ch_type ∗__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename

basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)

- template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type >
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits , typename _Ch_type >
  basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

2.36.1 Detailed Description

A facility for performing regular expression pattern matching.

2.36.2 Typedef Documentation

2.36.2.1 typedef regex_token_iterator<const char*> std::cregex_token_iterator

Token iterator for C-style NULL-terminated strings.

Definition at line 2469 of file regex.h.

2.36.2.2 typedef sub_match<const char*> std::csub_match

Standard regex submatch over a C-style null-terminated string.

Definition at line 834 of file regex.h.

2.36.2.3 typedef basic_regex<char> std::regex

Standard regular expressions.

Definition at line 706 of file regex.h.

2.36.2.4 typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator

Token iterator for standard strings.

Definition at line 2472 of file regex.h.

2.36.2.5 typedef sub_match<string::const_iterator> std::ssub_match

Standard regex submatch over a standard string.

Definition at line 837 of file regex.h.

2.36.2.6 typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2476 of file regex.h.

2.36.2.7 typedef sub_match<const wchar_t*> std::wcsub_match

Regex submatch over a C-style null-terminated wide string.

Definition at line 841 of file regex.h.

2.36.2.8 typedef basic_regex<wchar_t> std::wregex

Standard wide-character regular expressions.

Definition at line 710 of file regex.h.

2.36.2.9 typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator

Token iterator for standard wide-character strings.

Definition at line 2479 of file regex.h.

2.36.2.10 typedef sub_match<wstring::const_iterator> std::wssub_match

Regex submatch over a standard wide string.

Definition at line 844 of file regex.h.

2.36.3 Function Documentation

2.36.3.1 template<typename _Ch_type > bool std::regex_traits< _Ch_type >::isctype (_Ch_type __c, char_class_type __f) const

Determines if `c` is a member of an identified class.

Parameters

__c	a character.
__f	a class type (as returned from lookup_classname).

Returns

true if the character `__c` is a member of the classification represented by `__f`, false otherwise.

Exceptions

std::bad_cast	if the current locale does not have a ctype facet.

Definition at line 280 of file regex.h.

2.36.3.2 template<typename _BiIter > bool std::operator!= (const sub_match< _BiIter > & __lhs, const sub_match< _BiIter > & __rhs) `[inline]`

Tests the inequivalence of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if __*lhs* is not equivalent to __*rhs*, false otherwise.

Definition at line 868 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.3 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__lhs*, const sub_match< _Bi_iter > & *__rhs*) `[inline]`

Tests the inequivalence of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* is not equivalent to *__rhs*, false otherwise.

Definition at line 943 of file regex.h.

2.36.3.4 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator!= (const sub_match< _Bi_iter > & *__lhs*, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__rhs*) `[inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* is not equivalent to *__rhs*, false otherwise.

Definition at line 1017 of file regex.h.

2.36.3.5 template<typename _Bi_iter > bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const * *__lhs*, const sub_match< _Bi_iter > & *__rhs*) `[inline]`

Tests the inequivalence of an iterator value and a regular expression submatch.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* is not equivalent to *__rhs*, false otherwise.

Definition at line 1091 of file regex.h.

2.36.3.6 template<typename _Bi_iter > bool std::operator!= (const sub_match< _Bi_iter > & *__lhs*, typename iterator_traits< _Bi_iter >::value_type const * *__rhs*) `[inline]`

Tests the inequivalence of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A pointer to a string.

Returns

true if *__lhs* is not equivalent to *__rhs*, false otherwise.

Definition at line 1165 of file regex.h.

2.36.3.7 template< typename _Bi_iter > bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const & *__lhs,* const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the inequivalence of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* is not equivalent to *__rhs*, false otherwise.

Definition at line 1242 of file regex.h.

2.36.3.8 template< typename _Bi_iter > bool std::operator!= (const sub_match< _Bi_iter > & *__lhs,* typename iterator_traits< _Bi_iter >::value_type const & *__rhs*) [inline]

Tests the inequivalence of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A const string reference.

Returns

true if *__lhs* is not equivalent to *__rhs*, false otherwise.

Definition at line 1322 of file regex.h.

2.36.3.9 template< typename _Bi_iter , class _Alloc > bool std::operator!= (const match_results< _Bi_iter, _Alloc > & *__m1,* const match_results< _Bi_iter, _Alloc > & *__m2*) [inline]

Compares two match_results for inequality.

Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 1834 of file regex.h.

2.36.3.10 template< typename _BiIter > bool std::operator< (const sub_match< _BiIter > & *__lhs*, const sub_match< _BiIter > & *__rhs*) `[inline]`

Tests the ordering of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* precedes *__rhs*, false otherwise.

Definition at line 879 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.11 template< typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__lhs*, const sub_match< _Bi_iter > & *__rhs*) `[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* precedes *__rhs*, false otherwise.

Definition at line 955 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.12 template< typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator< (const sub_match< _Bi_iter > & *__lhs*, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__rhs*) `[inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if __*lhs* precedes __*rhs*, false otherwise.

Definition at line 1029 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

2.36.3.13 template< typename _Bi_iter > bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const * __*lhs*, const sub_match< _Bi_iter > & __*rhs*) `[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

__*lhs*	A string.
__*rhs*	A regular expression submatch.

Returns

true if __*lhs* precedes __*rhs*, false otherwise.

Definition at line 1103 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.14 template< typename _Bi_iter > bool std::operator< (const sub_match< _Bi_iter > & __*lhs*, typename iterator_traits< _Bi_iter >::value_type const * __*rhs*) `[inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

__*lhs*	A regular expression submatch.
__*rhs*	A string.

Returns

true if __*lhs* precedes __*rhs*, false otherwise.

Definition at line 1177 of file regex.h.

2.36.3.15 template< typename _Bi_iter > bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const & __*lhs*, const sub_match< _Bi_iter > & __*rhs*) `[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

__*lhs*	A string.
__*rhs*	A regular expression submatch.

Returns

true if __*lhs* precedes __*rhs*, false otherwise.

Definition at line 1254 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.16 template<typename _Bi_iter > bool std::operator< (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const & __rhs) [inline]

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A const string reference.

Returns

true if *__lhs* precedes *__rhs*, false otherwise.

Definition at line 1334 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.17 template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter > basic_ostream<_Ch_type, _Ch_traits>& std::operator<< (basic_ostream< _Ch_type, _Ch_traits > & __os, const sub_match< _Bi_iter > & __m) [inline]

Inserts a matched string into an output stream.

Parameters

__os	The output stream.
__m	A submatch string.

Returns

the output stream with the submatch string inserted.

Definition at line 1388 of file regex.h.

References std::sub_match< _BiIter >::str().

2.36.3.18 template<typename _BiIter > bool std::operator<= (const sub_match< _BiIter > & __lhs, const sub_match< _BiIter > & __rhs) [inline]

Tests the ordering of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 890 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.19 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__lhs*, const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 991 of file regex.h.

2.36.3.20 template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator<= (const sub_match< _Bi_iter > & *__lhs*, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__rhs*) [inline]

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 1065 of file regex.h.

2.36.3.21 template<typename _Bi_iter > bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const * *__lhs*, const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 1139 of file regex.h.

2.36.3.22 template<typename _Bi_iter > bool std::operator<= (const sub_match< _Bi_iter > & *__lhs*, typename iterator_traits< _Bi_iter >::value_type const * *__rhs*) [inline]

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 1213 of file regex.h.

2.36.3.23 template< typename _Bi_iter > bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const & *__lhs*, const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 1293 of file regex.h.

2.36.3.24 template< typename _Bi_iter > bool std::operator<= (const sub_match< _Bi_iter > & *__lhs*, typename iterator_traits< _Bi_iter >::value_type const & *__rhs*) [inline]

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A const string reference.

Returns

true if *__lhs* does not succeed *__rhs*, false otherwise.

Definition at line 1373 of file regex.h.

2.36.3.25 template< typename _BiIter > bool std::operator== (const sub_match< _BiIter > & *__lhs*, const sub_match< _BiIter > & *__rhs*) [inline]

Tests the equivalence of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

Definition at line 857 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.26 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__lhs*, const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the equivalence of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

Definition at line 930 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), and std::sub_match< _BiIter >::compare().

2.36.3.27 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator== (const sub_match< _Bi_iter > & *__lhs*, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__rhs*) [inline]

Tests the equivalence of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

Definition at line 1004 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), and std::sub_match< _BiIter >::compare().

2.36.3.28 template<typename _Bi_iter > bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const ∗ *__lhs*, const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the equivalence of a C string and a regular expression submatch.

Parameters

__lhs	A C string.
__rhs	A regular expression submatch.

Returns

true if __*lhs* is equivalent to __*rhs*, false otherwise.

Definition at line 1078 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.29 template<typename _Bi_iter > bool std::operator== (const sub_match< _Bi_iter > & __*lhs*, typename iterator_traits< _Bi_iter >::value_type const * __*rhs*) [inline]

Tests the equivalence of a regular expression submatch and a string.

Parameters

__*lhs*	A regular expression submatch.
__*rhs*	A pointer to a string?

Returns

true if __*lhs* is equivalent to __*rhs*, false otherwise.

Definition at line 1152 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.30 template<typename _Bi_iter > bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const & __*lhs*, const sub_match< _Bi_iter > & __*rhs*) [inline]

Tests the equivalence of a string and a regular expression submatch.

Parameters

__*lhs*	A string.
__*rhs*	A regular expression submatch.

Returns

true if __*lhs* is equivalent to __*rhs*, false otherwise.

Definition at line 1226 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.31 template<typename _Bi_iter > bool std::operator== (const sub_match< _Bi_iter > & __*lhs*, typename iterator_traits< _Bi_iter >::value_type const & __*rhs*) [inline]

Tests the equivalence of a regular expression submatch and a string.

Parameters

__*lhs*	A regular expression submatch.
__*rhs*	A const string reference.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

Definition at line 1306 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.32 template<typename _Bi_iter , typename _Alloc > bool std::operator== (const match_results< _Bi_iter, _Alloc > & *__m1,* const match_results< _Bi_iter, _Alloc > & *__m2*) [inline]

Compares two match_results for equality.

Returns

true if the two objects refer to the same match, false otherwise.

Definition at line 1810 of file regex.h.

References std::match_results< _Bi_iter, _Alloc >::begin(), std::match_results< _Bi_iter, _Alloc >::empty(), std::match_results< _Bi_iter, _Alloc >::end(), std::equal(), std::match_results< _Bi_iter, _Alloc >::prefix(), std::match_results< _Bi_iter, _Alloc >::ready(), std::match_results< _Bi_iter, _Alloc >::size(), and std::match_results< _Bi_iter, _Alloc >::suffix().

2.36.3.33 template<typename _BiIter > bool std::operator> (const sub_match< _BiIter > & *__lhs,* const sub_match< _BiIter > & *__rhs*) [inline]

Tests the ordering of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 912 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.34 template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__lhs,* const sub_match< _Bi_iter > & *__rhs*) [inline]

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 967 of file regex.h.

2.36.3.35 template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator> (const sub_match< _Bi_iter > & __lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs) [inline]

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 1041 of file regex.h.

2.36.3.36 template<typename _Bi_iter > bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const * __lhs, const sub_match< _Bi_iter > & __rhs) [inline]

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 1115 of file regex.h.

2.36.3.37 template<typename _Bi_iter > bool std::operator> (const sub_match< _Bi_iter > & __lhs, typename iterator_traits< _Bi_iter >::value_type const * __rhs) [inline]

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 1189 of file regex.h.

2.36.3.38 template<typename _Bi_iter > bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const & __lhs, const sub_match< _Bi_iter > & __rhs) [inline]

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 1269 of file regex.h.

2.36.3.39 template< typename _Bi_iter > bool std::operator> (const sub_match< _Bi_iter > & *__lhs,* typename iterator_traits< _Bi_iter >::value_type const & *__rhs*) `[inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A const string reference.

Returns

true if *__lhs* succeeds *__rhs*, false otherwise.

Definition at line 1349 of file regex.h.

2.36.3.40 template< typename _BiIter > bool std::operator>= (const sub_match< _BiIter > & *__lhs,* const sub_match< _BiIter > & *__rhs*) `[inline]`

Tests the ordering of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* does not precede *__rhs*, false otherwise.

Definition at line 901 of file regex.h.

References std::sub_match< _BiIter >::compare().

2.36.3.41 template< typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc > bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & *__lhs,* const sub_match< _Bi_iter > & *__rhs*) `[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

__lhs	A string.
__rhs	A regular expression submatch.

Returns

true if __*lhs* does not precede __*rhs*, false otherwise.

Definition at line 979 of file regex.h.

2.36.3.42 template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc > bool std::operator>= (const sub_match< _Bi_iter > & __*lhs*, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __*rhs*) `[inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

__*lhs*	A regular expression submatch.
__*rhs*	A string.

Returns

true if __*lhs* does not precede __*rhs*, false otherwise.

Definition at line 1053 of file regex.h.

2.36.3.43 template<typename _Bi_iter > bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const * __*lhs*, const sub_match< _Bi_iter > & __*rhs*) `[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

__*lhs*	A string.
__*rhs*	A regular expression submatch.

Returns

true if __*lhs* does not precede __*rhs*, false otherwise.

Definition at line 1127 of file regex.h.

2.36.3.44 template<typename _Bi_iter > bool std::operator>= (const sub_match< _Bi_iter > & __*lhs*, typename iterator_traits< _Bi_iter >::value_type const * __*rhs*) `[inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

__*lhs*	A regular expression submatch.
__*rhs*	A string.

Returns

true if __*lhs* does not precede __*rhs*, false otherwise.

Definition at line 1201 of file regex.h.

2.36.3.45 template< typename _Bi_iter > bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const & __*lhs*, const sub_match< _Bi_iter > & __*rhs*) `[inline]`

Tests the ordering of a string and a regular expression submatch.

Parameters

__*lhs*	A string.
__*rhs*	A regular expression submatch.

Returns

true if __*lhs* does not precede __*rhs*, false otherwise.

Definition at line 1281 of file regex.h.

2.36.3.46 template< typename _Bi_iter > bool std::operator>= (const sub_match< _Bi_iter > & __*lhs*, typename iterator_traits< _Bi_iter >::value_type const & __*rhs*) `[inline]`

Tests the ordering of a regular expression submatch and a string.

Parameters

__*lhs*	A regular expression submatch.
__*rhs*	A const string reference.

Returns

true if __*lhs* does not precede __*rhs*, false otherwise.

Definition at line 1361 of file regex.h.

2.36.3.47 template< typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits > bool std::regex_match (_Bi_iter __*s*, _Bi_iter __*e*, match_results< _Bi_iter, _Alloc > & __*m*, const basic_regex< _Ch_type, _Rx_traits > & __*re*, regex_constants::match_flag_type __*flags* = `regex_constants::match_default`)

Determines if there is a match between the regular expression `e` and all of the character sequence [first, last).

Parameters

__*s*	Start of the character sequence to match.
__*e*	One-past-the-end of the character sequence to match.
__*m*	The match results.
__*re*	The regular expression.
__*flags*	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

Todo Implement this function.

Definition at line 1878 of file regex.h.

Referenced by std::regex_match().

2.36.3.48 template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits > bool std::regex_match (_Bi_iter *__first,* _Bi_iter *__last,* const basic_regex< _Ch_type, _Rx_traits > & *__re,* regex_constants::match_flag_type *__flags =* `regex_constants::match_default`)

Indicates if there is a match between the regular expression `e` and all of the character sequence [first, last).

Parameters

__first	Beginning of the character sequence to match.
__last	One-past-the-end of the character sequence to match.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

Definition at line 1909 of file regex.h.

References std::regex_match().

2.36.3.49 template<typename _Ch_type , typename _Alloc , typename _Rx_traits > bool std::regex_match (const _Ch_type * *__s,* match_results< const _Ch_type *, _Alloc > & *__m,* const basic_regex< _Ch_type, _Rx_traits > & *__re,* regex_constants::match_flag_type *__f =* `regex_constants::match_default`) `[inline]`

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

__s	The C-style null-terminated string to match.
__m	The match results.
__re	The regular expression.
__f	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

Definition at line 1934 of file regex.h.

References std::regex_match().

2.36.3.50 template< typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits > bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & *__s*, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & *__m*, const basic_regex< _Ch_type, _Rx_traits > & *__re*, regex_constants::match_flag_type *__flags* = `regex_constants::match_default`) `[inline]`

Determines if there is a match between the regular expression `e` and a string.

Parameters

__s	The string to match.
__m	The match results.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

Definition at line 1958 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), and std::regex_match().

2.36.3.51 template< typename _Ch_type , class _Rx_traits > bool std::regex_match (const _Ch_type * *__s*, const basic_regex< _Ch_type, _Rx_traits > & *__re*, regex_constants::match_flag_type *__f* = `regex_constants::match_default`) `[inline]`

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

__s	The C-style null-terminated string to match.
__re	The regular expression.
__f	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

Definition at line 1981 of file regex.h.

References std::regex_match().

2.36.3.52 template<typename _Ch_traits , typename _Str_allocator , typename _Ch_type , typename _Rx_traits > bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __re, regex_constants::match_flag_type __flags = `regex_constants::match_default`) `[inline]`

Indicates if there is a match between the regular expression `e` and a string.

Parameters

__s	[IN] The string to match.
__re	[IN] The regular expression.
__flags	[IN] Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

Definition at line 2003 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), and std::regex_match().

2.36.3.53 template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type > _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > & __e, const basic_string< _Ch_type > & __fmt, regex_constants::match_flag_type __flags = `regex_constants::match_default`) `[inline]`

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Parameters

__out	
__first	
__last	
__e	
__fmt	
__flags	

Returns

out

Exceptions

an	exception of type regex_error.

Todo Implement this function.

Definition at line 2162 of file regex.h.

Referenced by std::regex_replace().

2.36.3.54 template<typename _Rx_traits , typename _Ch_type > basic_string<_Ch_type> std::regex_replace (const basic_string< _Ch_type > & *__s,* const basic_regex< _Ch_type, _Rx_traits > & *__e,* const basic_string< _Ch_type > & *__fmt,* regex_constants::match_flag_type *__flags* = `regex_constants::match_default`) `[inline]`

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html`

Parameters

__s	
__e	
__fmt	
__flags	

Returns

a copy of string `s` with replacements.

Exceptions

an	exception of type regex_error.

Definition at line 2182 of file regex.h.

References std::back_inserter(), std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), and std::regex_replace().

2.36.3.55 template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits > bool std::regex_search (_Bi_iter *__first,* _Bi_iter *__last,* match_results< _Bi_iter, _Alloc > & *__m,* const basic_regex< _Ch_type, _Rx_traits > & *__re,* regex_constants::match_flag_type *__flags* = `regex_constants::match_default`) `[inline]`

Searches for a regular expression within a range.

Parameters

__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__m	[OUT] The match results.
__re	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

Todo Implement this function.

Definition at line 2028 of file regex.h.

Referenced by std::regex_search().

2.36.3.56 template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits > bool std::regex_search (_Bi_iter *__first*, _Bi_iter *__last*, const basic_regex< _Ch_type, _Rx_traits > & *__re*, regex_constants::match_flag_type *__flags* = `regex_constants::match_default`) `[inline]`

Searches for a regular expression within a range.

Parameters

__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__re	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Exceptions

an	exception of type regex_error.

Definition at line 2049 of file regex.h.

References std::regex_search().

2.36.3.57 template<typename _Ch_type , class _Alloc , class _Rx_traits > bool std::regex_search (const _Ch_type * *__s*, match_results< const _Ch_type *, _Alloc > & *__m*, const basic_regex< _Ch_type, _Rx_traits > & *__e*, regex_constants::match_flag_type *__f* = `regex_constants::match_default`) `[inline]`

Searches for a regular expression within a C-string.

Parameters

__s	[IN] A C-string to search for the regex.
__m	[OUT] The set of regex matches.
__e	[IN] The regex to search for in `s`.
__f	[IN] The search flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Exceptions

an	exception of type regex_error.

Definition at line 2073 of file regex.h.

References std::regex_search().

2.36.3.58 template<typename _Ch_type , typename _Rx_traits > bool std::regex_search (const _Ch_type ∗ __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f = `regex_constants::match_default`) `[inline]`

Searches for a regular expression within a C-string.

Parameters

__s	[IN] The C-string to search.
__e	[IN] The regular expression to search for.
__f	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Exceptions

an	exception of type regex_error.

Definition at line 2093 of file regex.h.

References std::regex_search().

2.36.3.59 template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_traits > bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __flags = `regex_constants::match_default`) `[inline]`

Searches for a regular expression within a string.

Parameters

__s	[IN] The string to search.
__e	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Exceptions

an	exception of type regex_error.

Definition at line 2113 of file regex.h.

References std::regex_search().

2.36.3.60 template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits > bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m, const basic_regex< _Ch_type, _Rx_traits > & __e, regex_constants::match_flag_type __f = `regex_constants::match_default`) `[inline]`

Searches for a regular expression within a string.

Parameters

__s	[IN] A C++ string to search for the regex.
__m	[OUT] The set of regex matches.
__e	[IN] The regex to search for in `s`.
__f	[IN] The search flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

Definition at line 2136 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), and std::regex_search().

2.36.3.61 template<typename _Ch_type , typename _Rx_traits > void std::swap (basic_regex< _Ch_type, _Rx_traits > & __lhs, basic_regex< _Ch_type, _Rx_traits > & __rhs) `[inline]`

Swaps the contents of two regular expression objects.

Parameters

__lhs	First regular expression.
__rhs	Second regular expression.

Definition at line 722 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::swap().

2.36.3.62 template<typename _Bi_iter , typename _Alloc > void std::swap (match_results< _Bi_iter, _Alloc > & *__lhs,* match_results< _Bi_iter, _Alloc > & *__rhs*) [inline]

Swaps two match results.

Parameters

__lhs	A match result.
__rhs	A match result.

The contents of the two match_results objects are swapped.

Definition at line 1848 of file regex.h.

References std::match_results< _Bi_iter, _Alloc >::swap().

2.36.3.63 template<typename _Ch_type > int std::regex_traits< _Ch_type >::value (_Ch_type *__ch,* int *__radix*) const

Converts a digit to an int.

Parameters

__ch	a character representing a digit.
__radix	the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

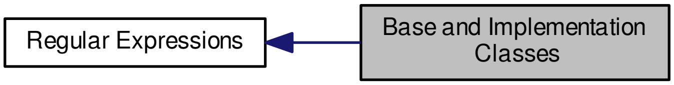
> the value represented by the digit __ch in base radix if the character __ch is a valid digit in base radix; otherwise returns -1.

Definition at line 314 of file regex.h.

References std::basic_ios< _CharT, _Traits >::fail(), std::hex(), and std::oct().

2.37 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- class std::__detail::_Automaton

  Base class for, um, automata. Could be an NFA or a DFA. Your choice.
- struct std::__detail::_CharMatcher< _InIterT, _TraitsT >

  Matches a single character.
- class std::__detail::_Compiler< _InIter, _TraitsT >

  Builds an NFA from an input iterator interval.
- struct std::__detail::_EndTagger< _FwdIterT, _TraitsT >

  End state tag.
- class std::__detail::_Grep_matcher

  Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.
- class std::__detail::_Nfa

  struct _Nfa
- struct std::__detail::_PatternCursor

  ABC for pattern matching.
- struct std::__detail::_RangeMatcher< _InIterT, _TraitsT >

  Matches a character range (bracket expression)
- struct std::__detail::_Results

  Provides a generic facade for a templated match_results.
- class std::__detail::_Scanner< _InputIterator >

  struct _Scanner. Scans an input range for regex tokens.
- struct std::__detail::_Scanner_base

  Base class for scanner.
- class std::__detail::_SpecializedCursor< _FwdIterT >

  Provides a cursor into the specific target string.
- class std::__detail::_SpecializedResults< _FwdIterT, _Alloc >

  A _Results facade specialized for wrapping a templated match_results.
- struct std::__detail::_StartTagger< _FwdIterT, _TraitsT >

  Start state tag.
- struct std::__detail::_State

  struct _State
- class std::__detail::_StateSeq

  Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Typedefs

- typedef std::shared_ptr
  < _Automaton > std::__detail::_AutomatonPtr
- typedef std::function< bool(const
  _PatternCursor &)> std::__detail::_Matcher
- typedef int std::__detail::_StateIdT
- typedef std::set< _StateIdT > std::__detail::_StateSet
- typedef std::stack< _StateIdT,
  std::vector< _StateIdT > > std::__detail::_StateStack
- typedef std::function< void(const
  _PatternCursor &, _Results &)> std::__detail::_Tagger

Enumerations

- enum std::__detail::_Opcode {
  _S_opcode_unknown, **_S_opcode_alternative**, **_S_opcode_subexpr_begin**, **_S_opcode_subexpr_end**, **_S_opcode_match**, **_S_opcode_accept** }

Functions

- **std::__detail::_Compiler< _InIter, _TraitsT >::_Compiler** (const _InIter &__b, const _InIter &__e, _TraitsT &__traits, _FlagT __flags)
- **std::__detail::_SpecializedResults< _FwdIterT, _Alloc >::_SpecializedResults** (const _Automaton::_SizeT __size, const _SpecializedCursor< _FwdIterT > &__cursor, match_results< _FwdIterT, _Alloc > &__m)
- template<typename _InIter , typename _TraitsT >
  _AutomatonPtr **std::__detail::__compile** (const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)
- template<typename _FwdIterT >
  _SpecializedCursor< _FwdIterT > **std::__detail::__cursor** (const _FwdIterT &__b, const _FwdIterT __e)
- bool std::__detail::_AnyMatcher (const _PatternCursor &)
- void **std::__detail::_Scanner< _InputIterator >::_M_advance** ()
- void **std::__detail::_SpecializedResults< _FwdIterT, _Alloc >::_M_set_pos** (int __i, int __j, const _PatternCursor &__pc)

Variables

- static const _StateIdT std::__detail::_S_invalid_state_id

2.37.1 Detailed Description

2.37.2 Typedef Documentation

2.37.2.1 typedef std::shared_ptr<_Automaton> std::__detail::_AutomatonPtr

Generic shared pointer to an automaton.

Definition at line 62 of file regex_nfa.h.

2.37.2.2 typedef std::function<bool (const _PatternCursor&)> std::__detail::_Matcher

Indicates if current state matches cursor current.

Definition at line 120 of file regex_nfa.h.

2.37.2.3 typedef int std::__detail::_StateIdT

Identifies a state in the NFA.

Definition at line 195 of file regex_nfa.h.

2.37.2.4 typedef std::set<_StateIdT> std::__detail::_StateSet

The Grep Matcher works on sets of states. Here are sets of states.

Definition at line 251 of file regex_nfa.h.

2.37.2.5 typedef std::stack<_StateIdT, std::vector<_StateIdT> > std::__detail::_StateStack

A stack of states used in evaluating the NFA.

Definition at line 105 of file regex_grep_matcher.h.

2.37.2.6 typedef std::function<void (const _PatternCursor&, _Results&)> std::__detail::_Tagger

Tags current state (for subexpr begin/end).

Definition at line 84 of file regex_nfa.h.

2.37.3 Enumeration Type Documentation

2.37.3.1 enum std::__detail::_Opcode

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

Definition at line 66 of file regex_nfa.h.

2.37.4 Function Documentation

2.37.4.1 bool std::__detail::_AnyMatcher (const _PatternCursor &) `[inline]`

Matches any character.

Definition at line 124 of file regex_nfa.h.

2.37.5 Variable Documentation

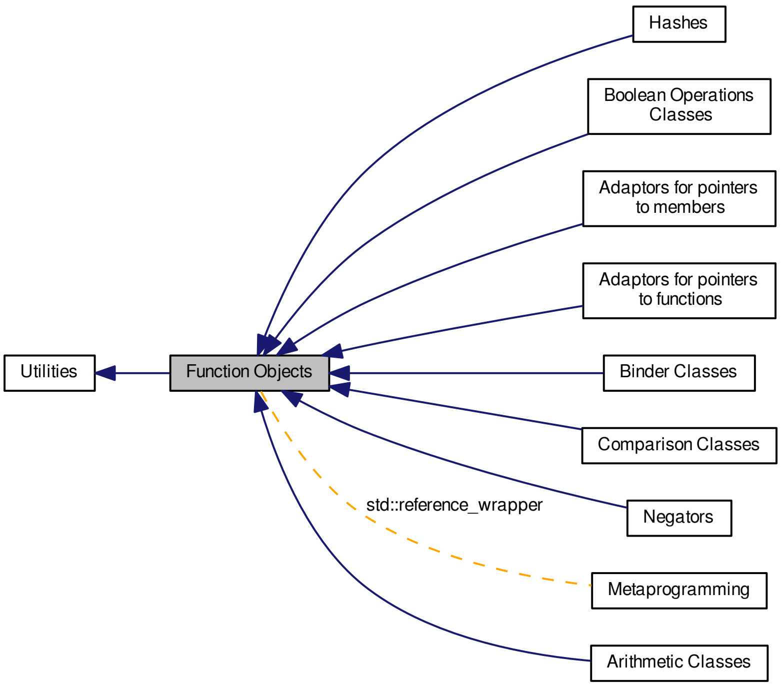
2.37.5.1 const _StateIdT std::__detail::_S_invalid_state_id `[static]`

The special case in which a state identifier is not an index.

Definition at line 198 of file regex_nfa.h.

2.38 Function Objects

Collaboration diagram for Function Objects:



Modules

- Adaptors for pointers to functions
- Adaptors for pointers to members
- Arithmetic Classes
- Binder Classes
- Boolean Operations Classes
- Comparison Classes
- Hashes
- Negators

Classes

- struct std::binary_function< _Arg1, _Arg2, _Result >
- class std::function< _Res(_ArgTypes...)>

  Primary class template for std::function.
  Polymorphic function wrapper.
- class std::reference_wrapper< _Tp >

  Primary class template for reference_wrapper.
- struct std::unary_function< _Arg, _Result >

Functions

- template<typename _Tp , typename _Class >
  _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept

2.38.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

2.38.2 Function Documentation

2.38.2.1 template<typename _Tp , typename _Class > _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::* *__pm*) `[inline]`, `[noexcept]`

Returns a function object that forwards to the member pointer *pm*.

Definition at line 961 of file functional.

2.39 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:

Classes

- struct std::divides< _Tp >

    One of the math functors.
- struct std::minus< _Tp >

    One of the math functors.
- struct std::modulus< _Tp >

    One of the math functors.
- struct std::multiplies< _Tp >

    One of the math functors.
- struct std::negate< _Tp >

    One of the math functors.
- struct std::plus< _Tp >

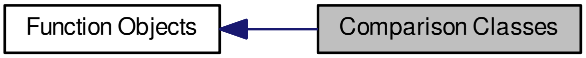
    One of the math functors.

2.39.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for the base classes for examples of their use.

2.40 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct std::equal_to< _Tp >

  One of the comparison functors.
- struct std::greater< _Tp >

  One of the comparison functors.
- struct std::greater_equal< _Tp >

  One of the comparison functors.
- struct std::less< _Tp >

  One of the comparison functors.
- struct std::less_equal< _Tp >

  One of the comparison functors.
- struct std::not_equal_to< _Tp >

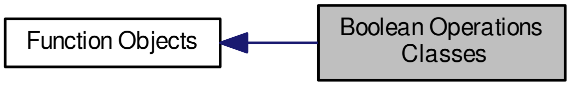
  One of the comparison functors.

2.40.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like $<$.

2.41 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

- struct std::logical_and< _Tp >

  One of the Boolean operations functors.
- struct std::logical_not< _Tp >

  One of the Boolean operations functors.
- struct std::logical_or< _Tp >

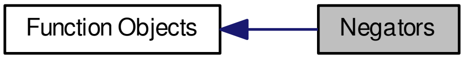
  One of the Boolean operations functors.

2.41.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, ||, and `!`.

2.42 Negators

Collaboration diagram for Negators:



Classes

- class std::binary_negate< _Predicate >

  One of the negation functors.
- class std::unary_negate< _Predicate >

  One of the negation functors.

Functions

- template<typename _Predicate >
  unary_negate< _Predicate > std::not1 (const _Predicate &__pred)
- template<typename _Predicate >
  binary_negate< _Predicate > std::not2 (const _Predicate &__pred)

2.42.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```
struct IntGreaterThanThree
  : public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (i) of `v` for which `!(v[i] > 3)` is true.

The not1/unary_negate combination works on predicates taking a single argument. The not2/binary_negate combination works on predicates which take two arguments.

2.42.2 Function Documentation

2.42.2.1 template<typename _Predicate > unary_negate<_Predicate> std::not1 (const _Predicate & *__pred*) `[inline]`

One of the negation functors.

Definition at line 369 of file stl_function.h.

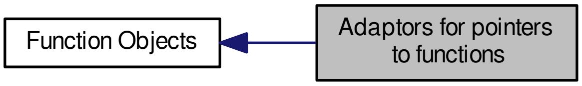
2.42.2.2 template<typename _Predicate > binary_negate<_Predicate> std::not2 (const _Predicate & *__pred*) [inline]

One of the negation functors.

Definition at line 394 of file stl_function.h.

2.43 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >

  One of the adaptors for function pointers.
- class std::pointer_to_unary_function< _Arg, _Result >

  One of the adaptors for function pointers.

Functions

- template<typename _Arg , typename _Result >
  pointer_to_unary_function
  < _Arg, _Result > std::ptr_fun (_Result($*$__x)(_Arg))
- template<typename _Arg1 , typename _Arg2 , typename _Result >
  pointer_to_binary_function
  < _Arg1, _Arg2, _Result > std::ptr_fun (_Result($*$__x)(_Arg1, _Arg2))

2.43.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

2.43.2 Function Documentation

2.43.2.1 template<typename _Arg , typename _Result > pointer_to_unary_function<_Arg, _Result> std::ptr_fun (_Result($*$)(_Arg) __x) `[inline]`

One of the adaptors for function pointers.

Definition at line 442 of file stl_function.h.

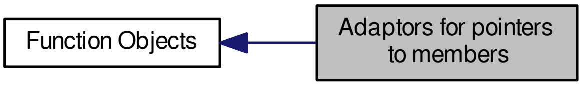
2.43.2.2 template<typename _Arg1 , typename _Arg2 , typename _Result > pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (_Result(*)(_Arg1, _Arg2) __x) `[inline]`

One of the adaptors for function pointers.

Definition at line 468 of file stl_function.h.

2.44 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::const_mem_fun1_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::const_mem_fun_ref_t< _Ret, _Tp >

  One of the adaptors for member pointers.
- class std::const_mem_fun_t< _Ret, _Tp >

  One of the adaptors for member pointers.
- class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::mem_fun1_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::mem_fun_ref_t< _Ret, _Tp >

  One of the adaptors for member pointers.
- class std::mem_fun_t< _Ret, _Tp >

  One of the adaptors for member pointers.

Functions

- template<typename _Ret , typename _Tp >
  mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::∗__f)())
- template<typename _Ret , typename _Tp , typename _Arg >
  mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::∗__f)(_Arg))
- template<typename _Ret , typename _Tp >
  mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::∗__f)())
- template<typename _Ret , typename _Tp , typename _Arg >
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::∗__f)(_Arg))

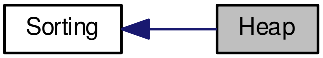
2.44.1 Detailed Description

There are a total of 8 = 2^3 function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function mem_fun and mem_fun_ref, which create whichever type of adaptor is appropriate.

2.45 Heap

Collaboration diagram for Heap:



Functions

- template<typename _RandomAccessIterator >
  bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)

2.45.1 Detailed Description

2.45.2 Function Documentation

2.45.2.1 template<typename _RandomAccessIterator > bool std::is_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) `[inline]`

Determines whether a range is a heap.

Parameters

__first	Start of range.
__last	End of range.

Returns

True if range is a heap, false otherwise.

Definition at line 565 of file stl_heap.h.

References std::is_heap_until().

2.45.2.2 template<typename _RandomAccessIterator , typename _Compare > bool std::is_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*) [inline]

Determines whether a range is a heap using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor to use.

Returns

True if range is a heap, false otherwise.

Definition at line 578 of file stl_heap.h.

References std::is_heap_until().

2.45.2.3 template<typename _RandomAccessIterator > _RandomAccessIterator std::is_heap_until (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) [inline]

Search the end of a heap.

Parameters

__first	Start of range.
__last	End of range.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator i in [__first, __last) for which the range [__first, i) is a heap.

Definition at line 517 of file stl_heap.h.

References std::distance().

2.45.2.4 template<typename _RandomAccessIterator , typename _Compare > _RandomAccessIterator std::is_heap_until (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*) [inline]

Search the end of a heap using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor to use.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator i in [__first, __last) for which the range [__first, i) is a heap. Comparisons are made using __comp.

Definition at line 543 of file stl_heap.h.

References std::distance().

Referenced by std::is_heap().

2.45.2.5 template<typename _RandomAccessIterator > void std::make_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*)

Construct a heap over a range.

Parameters

__first	Start of heap.
__last	End of heap.

This operation makes the elements in [__first,__last) into a heap.

Definition at line 380 of file stl_heap.h.

2.45.2.6 template<typename _RandomAccessIterator , typename _Compare > void std::make_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*)

Construct a heap over a range using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap.
__comp	Comparison functor to use.

This operation makes the elements in [__first,__last) into a heap. Comparisons are made using __comp.

Definition at line 420 of file stl_heap.h.

Referenced by std::__heap_select(), std::partial_sort_copy(), and std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue().

2.45.2.7 template<typename _RandomAccessIterator > void std::pop_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) `[inline]`

Pop an element off a heap.

Parameters

__first	Start of heap.
__last	End of heap.

Precondition

[__first, __last) is a valid, non-empty range.

This operation pops the top of the heap. The elements __first and __last-1 are swapped and [__first,__last-1) is made into a heap.

Definition at line 281 of file stl_heap.h.

2.45.2.8 template<typename _RandomAccessIterator , typename _Compare > void std::pop_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*) `[inline]`

Pop an element off a heap using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap.
__comp	Comparison functor to use.

This operation pops the top of the heap. The elements __first and __last-1 are swapped and [__first,__last-1) is made into a heap. Comparisons are made using comp.

Definition at line 356 of file stl_heap.h.

Referenced by std::priority_queue< _Tp, _Sequence, _Compare >::pop().

2.45.2.9 template<typename _RandomAccessIterator > void std::push_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*) `[inline]`

Push an element onto a heap.

Parameters

__first	Start of heap.
__last	End of heap + element.

This operation pushes the element at last-1 onto the valid heap over the range [__first,__last-1). After completion, [__first,__last) is a valid heap.

Definition at line 156 of file stl_heap.h.

2.45.2.10 template<typename _RandomAccessIterator , typename _Compare > void std::push_heap (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*) `[inline]`

Push an element onto a heap using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap + element.
__comp	Comparison functor.

This operation pushes the element at __last-1 onto the valid heap over the range [__first,__last-1). After completion, [__first,__last) is a valid heap. Compare operations are performed using comp.

Definition at line 206 of file stl_heap.h.

Referenced by std::priority_queue< _Tp, _Sequence, _Compare >::push().

2.45.2.11 template<typename _RandomAccessIterator > void std::sort_heap (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*)

Sort a heap.

Parameters

__first	Start of heap.
__last	End of heap.

This operation sorts the valid heap in the range [__first,__last).

Definition at line 459 of file stl_heap.h.

2.45.2.12 template<typename _RandomAccessIterator , typename _Compare > void std::sort_heap (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*, _Compare *__comp*)

Sort a heap using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap.
__comp	Comparison functor to use.

This operation sorts the valid heap in the range [__first,__last). Comparisons are made using __comp.

Definition at line 488 of file stl_heap.h.

Referenced by std::partial_sort(), and std::partial_sort_copy().

2.46 Iterators

Collaboration diagram for Iterators:

Modules

- Iterator Tags

Classes

- class std::__has_iterator_category_helper< _Tp >

  Traits class for iterators.
- class std::back_insert_iterator< _Container >

  Turns assignment into insertion.
- class std::front_insert_iterator< _Container >

  Turns assignment into insertion.
- struct std::input_iterator_tag

  Marking input iterators.
- class std::insert_iterator< _Container >

  Turns assignment into insertion.
- class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >

  Provides input iterator semantics for streams.
- class std::istreambuf_iterator< _CharT, _Traits >

  Provides input iterator semantics for streambufs.
- struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >

  Common iterator class.
- struct std::iterator_traits< _Tp $*$ >

  Partial specialization for pointer types.
- struct std::iterator_traits< const _Tp $*$ >

  Partial specialization for const pointer types.
- class std::move_iterator< _Iterator >
- class std::ostream_iterator< _Tp, _CharT, _Traits >

  Provides output iterator semantics for streams.
- class std::ostreambuf_iterator< _CharT, _Traits >

  Provides output iterator semantics for streambufs.
- class std::reverse_iterator< _Iterator >

Functions

- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _Iter >
  iterator_traits< _Iter >
  ::iterator_category std::__iterator_category (const _Iter &)
- template<typename _Iterator , typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>
  _ReturnType **std::__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Container >
  back_insert_iterator< _Container > std::back_inserter (_Container &__x)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  istreambuf_iterator< _CharT >
  >::__type **std::find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _Container >
  front_insert_iterator< _Container > std::front_inserter (_Container &__x)
- template<typename _Container , typename _Iterator >
  insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)
- template<typename _Iterator >
  move_iterator< _Iterator > **std::make_move_iterator** (_Iterator __i)
- template<class _Tp , class _CharT , class _Traits , class _Dist >
  bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)
- template<typename _CharT , typename _Traits >
  bool **std::operator!=** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _Iterator >
  bool **std::operator!=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)

- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator!=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator!=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  reverse_iterator< _Iterator > **std::operator+** (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _Iterator >
  move_iterator< _Iterator > **std::operator+** (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator >
  reverse_iterator< _Iterator >
  ::difference_type **std::operator-** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  auto **std::operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base()-__x.base())
- template<typename _IteratorL , typename _IteratorR >
  auto **std::operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(__x.base()-__y.base())
- template<typename _Iterator >
  auto **std::operator-** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) -> decltype(__x.base()-__y.base())
- template<typename _Iterator >
  bool **std::operator<** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator<** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  bool **std::operator<=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator<=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Tp , typename _CharT , typename _Traits , typename _Dist >
  bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)
- template<typename _CharT , typename _Traits >
  bool **std::operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _Iterator >
  bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator==** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)

- template<typename _Iterator >
  bool **std::operator>** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator>** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  bool **std::operator>=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator>=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)

2.46.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

2.46.2 Function Documentation

2.46.2.1 template<typename _Iter > iterator_traits<_Iter>::iterator_category std::__iterator_category (const _Iter &) [inline]

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 201 of file stl_iterator_base_types.h.

Referenced by std::__find_if_not(), __gnu_debug::__valid_range_aux(), std::advance(), __gnu_cxx::copy_n(), std::copy_n(), __gnu_cxx::distance(), std::distance(), std::find(), std::find_end(), std::find_if(), std::partition(), std::reverse(), std::search_n(), __gnu_cxx::uninitialized_copy_n(), std::uninitialized_copy_n(), and std::unique_copy().

2.46.2.2 template<typename _Container > back_insert_iterator<_Container> std::back_inserter (_Container & *__x*) [inline]

Parameters

__x	A container of arbitrary type.

Returns

An instance of back_insert_iterator working on `__x`.

This wrapper function helps in creating back_insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 479 of file stl_iterator.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::format(), and std::regex_replace().

2.46.2.3 template<typename _Container > front_insert_iterator<_Container> std::front_inserter (_Container & *__x*) [inline]

Parameters

__x	A container of arbitrary type.

Returns

An instance of front_insert_iterator working on `x`.

This wrapper function helps in creating front_insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 569 of file stl_iterator.h.

2.46.2.4 template<typename _Container , typename _Iterator > insert_iterator<_Container> std::inserter (_Container & *__x*, _Iterator *__i*) `[inline]`

Parameters

__x	A container of arbitrary type.

Returns

An instance of insert_iterator working on `__x`.

This wrapper function helps in creating insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 683 of file stl_iterator.h.

2.46.2.5 template<class _Tp , class _CharT , class _Traits , class _Dist > bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > & *__x*, const istream_iterator< _Tp, _CharT, _Traits, _Dist > & *__y*) `[inline]`

Return false if x and y are both end or not end, or x and y are the same.

Definition at line 137 of file stream_iterator.h.

2.46.2.6 template<typename _Tp , typename _CharT , typename _Traits , typename _Dist > bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > & *__x*, const istream_iterator< _Tp, _CharT, _Traits, _Dist > & *__y*) `[inline]`

Return true if x and y are both end or not end, or x and y are the same.

Definition at line 130 of file stream_iterator.h.

2.46.2.7 template<typename _Iterator > bool std::operator== (const reverse_iterator< _Iterator > & *__x*, const reverse_iterator< _Iterator > & *__y*) `[inline]`

Parameters

__x	A reverse_iterator.
__y	A reverse_iterator.

Returns

A simple bool.

Reverse iterators forward many operations to their underlying base() iterators. Others are implemented in terms of one another.

Definition at line 291 of file stl_iterator.h.

References std::reverse_iterator< _Iterator >::base().

2.47 Iterator Tags

Collaboration diagram for Iterator Tags:

Classes

- struct std::bidirectional_iterator_tag

  Bidirectional iterators support a superset of forward iterator operations.
- struct std::forward_iterator_tag

  Forward iterators support a superset of input iterator operations.
- struct std::input_iterator_tag

  Marking input iterators.
- struct std::output_iterator_tag

  Marking output iterators.
- struct std::random_access_iterator_tag

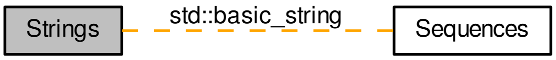
  Random-access iterators support a superset of bidirectional iterator operations.

2.47.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

2.48 Strings

Collaboration diagram for Strings:



Classes

- class std::basic_string< _CharT, _Traits, _Alloc >

  Managing sequences of characters and character-like objects.
- struct std::char_traits< _CharT >

  Basis for explicit traits specializations.

Typedefs

- typedef basic_string< char > std::string
- typedef basic_string< char16_t > std::u16string
- typedef basic_string< char32_t > std::u32string
- typedef basic_string< wchar_t > std::wstring

2.48.1 Detailed Description

2.48.2 Typedef Documentation

2.48.2.1 typedef basic_string<char> **std::string**

A string of `char`.

Definition at line 64 of file stringfwd.h.

2.48.2.2 typedef basic_string<char16_t> **std::u16string**

A string of `char16_t`.

Definition at line 80 of file stringfwd.h.

2.48.2.3 typedef basic_string<char32_t> **std::u32string**

A string of `char32_t`.

Definition at line 83 of file stringfwd.h.

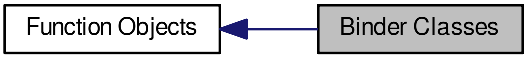
2.48.2.4 typedef basic_string<wchar_t> **std::wstring**

A string of `wchar_t`.

Definition at line 70 of file stringfwd.h.

2.49 Binder Classes

Collaboration diagram for Binder Classes:



Namespaces

- namespace std::placeholders

Classes

- struct std::_Placeholder< _Num >

  The type of placeholder objects defined by libstdc++.
- class std::binder1st< _Operation >

  One of the binder functors.
- class std::binder2nd< _Operation >

  One of the binder functors.
- struct std::is_bind_expression< _Tp >

  Determines if the given type _Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].
- struct std::is_bind_expression< _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_bind_expression< const _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_bind_expression< const volatile _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_bind_expression< volatile _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_placeholder< _Tp >

  Determines if the given type _Tp is a placeholder in a bind() expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct std::is_placeholder< _Placeholder< _Num > >

Functions

- template<typename _Func , typename... _BoundArgs>
  _Bind_helper< __is_socketlike
  < _Func >::value, _Func,
  _BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&...__args)
- template<typename _Result , typename _Func , typename... _BoundArgs>
  _Bindres_helper< _Result,
  _Func, _BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&...__args)
- template<typename _Operation , typename _Tp >
  binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation , typename _Tp >
  binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)

2.49.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, B's `operator()` is called with a single argument `y`. The return value is the value of `f(x,y)`. `B` can be *called* with various arguments (y1, y2, ...) and will in turn call `f(x,y1)`, `f(x,y2)`, ...

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd`(std::minus<float>(),1.3) will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform *1.3 - x* instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st`(std::plus<int>(),5).

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::placeholders::_1, x)`.

2.49.2 Function Documentation

2.49.2.1 template<typename _Func , typename... _BoundArgs> _Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs...>::type std::bind (_Func && __*f*, _BoundArgs &&... __*args*) [inline]

Function template for std::bind.

Definition at line 1655 of file functional.

Referenced by std::is_permutation().

2.49.2.2 template<typename _Result , typename _Func , typename... _BoundArgs> _Bindres_helper<_Result, _Func, _BoundArgs...>::type std::bind (_Func && __*f*, _BoundArgs &&... __*args*) [inline]

Function template for std::bind<R>.

Definition at line 1682 of file functional.

2.49.2.3 template<typename _Operation , typename _Tp > binder1st<_Operation> std::bind1st (const _Operation & *__fn*, const _Tp & *__x*) `[inline]`

One of the binder functors.

Definition at line 131 of file binders.h.

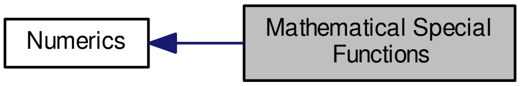
2.49.2.4 template<typename _Operation , typename _Tp > binder2nd<_Operation> std::bind2nd (const _Operation & *__fn*, const _Tp & *__x*) `[inline]`

One of the binder functors.

Definition at line 166 of file binders.h.

2.50 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx , typename _Tpy >
  __gnu_cxx::__promote_2< _Tpx,
  _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp , typename _Tpn >
  __gnu_cxx::__promote_2< _Tp,
  _Tpn >::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa , typename _Tpc , typename _Tp >
  __gnu_cxx::__promote_3< _Tpa,
  _Tpc, _Tp >::__type std::tr1::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float **std::tr1::conf_hypergf** (float __a, float __c, float __x)
- long double **std::tr1::conf_hypergl** (long double __a, long double __c, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)

- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp , typename _Tpp >
  __gnu_cxx::__promote_2< _Tp,
  _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp , typename _Tpp >
  __gnu_cxx::__promote_2< _Tp,
  _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp , typename _Tpn , typename _Tpp >
  __gnu_cxx::__promote_3< _Tp,
  _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
  __gnu_cxx::__promote_4< _Tpa,
  _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **std::tr1::hypergf** (float __a, float __b, float __c, float __x)
- long double **std::tr1::hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)

- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

2.50.1 Detailed Description

A collection of advanced mathematical special functions.

2.50.2 Function Documentation

2.50.2.1 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_laguerre (unsigned int *__n*, unsigned int *__m*, _Tp *__x*) `[inline]`

5.2.1.1 Associated Laguerre polynomials.

Definition at line 1055 of file tr1/cmath.

2.50.2.2 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_legendre (unsigned int *__l*, unsigned int *__m*, _Tp *__x*) `[inline]`

5.2.1.2 Associated Legendre functions.

Definition at line 1072 of file tr1/cmath.

2.50.2.3 template<typename _Tpx , typename _Tpy > __gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta (_Tpx *__x*, _Tpy *__y*) `[inline]`

5.2.1.3 Beta functions.

Definition at line 1089 of file tr1/cmath.

2.50.2.4 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_1 (_Tp *__k*) `[inline]`

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 1106 of file tr1/cmath.

2.50.2.5 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_2 (_Tp *__k*) `[inline]`

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 1123 of file tr1/cmath.

2.50.2.6 template<typename _Tp , typename _Tpn > __gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3 (_Tp *__k*, _Tpn *__nu*) [inline]

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 1140 of file tr1/cmath.

2.50.2.7 template<typename _Tpa , typename _Tpc , typename _Tp > __gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type std::tr1::conf_hyperg (_Tpa *__a*, _Tpc *__c*, _Tp *__x*) [inline]

5.2.1.7 Confluent hypergeometric functions.

Definition at line 1157 of file tr1/cmath.

2.50.2.8 template<typename _Tpnu , typename _Tp > __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i (_Tpnu *__nu*, _Tp *__x*) [inline]

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 1174 of file tr1/cmath.

2.50.2.9 template<typename _Tpnu , typename _Tp > __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j (_Tpnu *__nu*, _Tp *__x*) [inline]

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 1191 of file tr1/cmath.

2.50.2.10 template<typename _Tpnu , typename _Tp > __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_k (_Tpnu *__nu*, _Tp *__x*) [inline]

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 1208 of file tr1/cmath.

2.50.2.11 template<typename _Tpnu , typename _Tp > __gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_neumann (_Tpnu *__nu*, _Tp *__x*) [inline]

5.2.1.11 Cylindrical Neumann functions.

Definition at line 1225 of file tr1/cmath.

2.50.2.12 template<typename _Tp , typename _Tpp > __gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (_Tp *__k*, _Tpp *__phi*) [inline]

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 1242 of file tr1/cmath.

2.50.2.13 template<typename _Tp , typename _Tpp > __gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (_Tp *__k*, _Tpp *__phi*) [inline]

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 1259 of file tr1/cmath.

2.50.2.14 template<typename _Tp , typename _Tpn , typename _Tpp > __gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::tr1::ellint_3 (_Tp *__k*, _Tpn *__nu*, _Tpp *__phi*) [inline]

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 1276 of file tr1/cmath.

2.50.2.15 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::expint (_Tp *__x*) [inline]

5.2.1.15 Exponential integrals.

Definition at line 1293 of file tr1/cmath.

2.50.2.16 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::hermite (unsigned int *__n*, _Tp *__x*) [inline]

5.2.1.16 Hermite polynomials.

Definition at line 1310 of file tr1/cmath.

2.50.2.17 template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp > __gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type std::tr1::hyperg (_Tpa *__a*, _Tpb *__b*, _Tpc *__c*, _Tp *__x*) [inline]

5.2.1.17 Hypergeometric functions.

Definition at line 1327 of file tr1/cmath.

2.50.2.18 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::laguerre (unsigned int *__n*, _Tp *__x*) [inline]

5.2.1.18 Laguerre polynomials.

Definition at line 1344 of file tr1/cmath.

2.50.2.19 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::legendre (unsigned int *__n*, _Tp *__x*) [inline]

5.2.1.19 Legendre polynomials.

Definition at line 1361 of file tr1/cmath.

2.50.2.20 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::riemann_zeta (_Tp *__x*) [inline]

5.2.1.20 Riemann zeta function.

Definition at line 1378 of file tr1/cmath.

2.50.2.21 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_bessel (unsigned int *__n*, _Tp *__x*) [inline]

5.2.1.21 Spherical Bessel functions.

Definition at line 1395 of file tr1/cmath.

2.50.2.22 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_legendre (unsigned int *__l*, unsigned int *__m*, _Tp *__theta*) [inline]

5.2.1.22 Spherical associated Legendre functions.

Definition at line 1412 of file tr1/cmath.

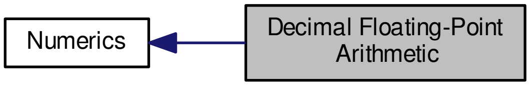
2.50.2.23 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::tr1::sph_neumann (unsigned int *__n*, _Tp *__x*) [inline]

5.2.1.23 Spherical Neumann functions.

Definition at line 1429 of file tr1/cmath.

2.51 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

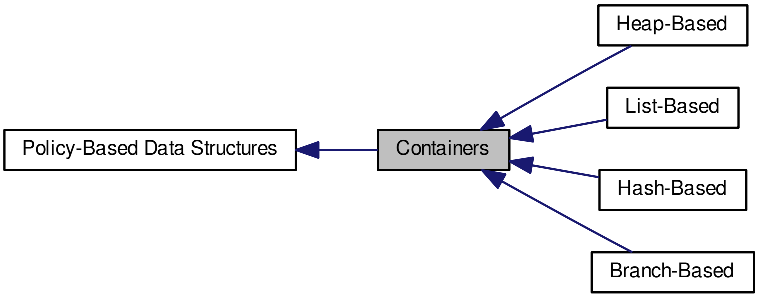
- namespace std::decimal

2.51.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

2.52 Containers

Collaboration diagram for Containers:



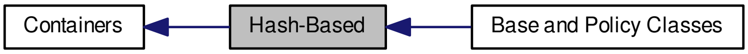
Modules

- Branch-Based
- Hash-Based
- Heap-Based
- List-Based

2.52.1 Detailed Description

2.53 Hash-Based

Collaboration diagram for Hash-Based:



Modules

- Base and Policy Classes

Classes

- class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
- class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >

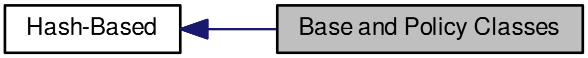
Macros

- #define **PB_DS_CC_HASH_BASE**
- #define **PB_DS_GP_HASH_BASE**
- #define **PB_DS_HASH_BASE**

2.53.1 Detailed Description

2.54 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



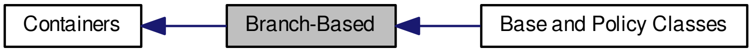
Classes

- class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >
- class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

2.54.1 Detailed Description

2.55 Branch-Based

Collaboration diagram for Branch-Based:



Modules

- Base and Policy Classes

Classes

- class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

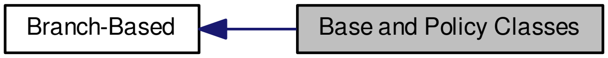
Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

2.55.1 Detailed Description

2.56 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



Classes

- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

  Ordered-vector tree associative-container.
- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >

  Conditional destructor.
- class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

  PATRICIA trie.
  This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.
- class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

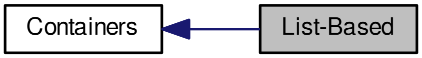
  Red-Black tree.
  This implementation uses an idea from the SGI STL (using a header node which is needed for efficient iteration).
- class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

  Splay tree.

2.56.1 Detailed Description

2.57 List-Based

Collaboration diagram for List-Based:



Classes

- class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >

Macros

- #define **PB_DS_LU_BASE**

2.57.1 Detailed Description

2.58 Exceptions

Collaboration diagram for Exceptions:

Classes

- struct __gnu_pbds::container_error

  Base class for exceptions.
- struct __gnu_pbds::insert_error

  An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unabvailable, in which case the allocator_type's exception will be thrown).
- struct __gnu_pbds::join_error

  A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.
- struct __gnu_pbds::resize_error

  A container cannot be resized.

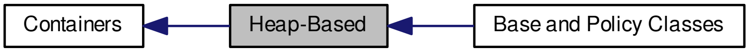
Functions

- void **__gnu_pbds::__throw_container_error** ()
- void **__gnu_pbds::__throw_insert_error** ()
- void **__gnu_pbds::__throw_join_error** ()
- void **__gnu_pbds::__throw_resize_error** ()

2.58.1 Detailed Description

2.59 Heap-Based

Collaboration diagram for Heap-Based:



Modules

- Base and Policy Classes

Classes

- class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

Typedefs

- typedef _Alloc **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::allocator_type**
- typedef Cmp_Fn **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::cmp_fn**
- typedef base_type::const_iterator **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::const_iterator**
- typedef __rebind_va::const_pointer **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::const_pointer**
- typedef __rebind_va::const_reference **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::const_reference**
- typedef Tag **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::container_category**
- typedef allocator_type::difference_type **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::difference_type**
- typedef base_type::iterator **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::iterator**
- typedef base_type::point_const_iterator **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::point_const_iterator**
- typedef base_type::point_iterator **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::point_iterator**
- typedef __rebind_va::pointer **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::pointer**
- typedef __rebind_va::reference **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::reference**
- typedef allocator_type::size_type **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::size_type**
- typedef _Tv **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::value_type**

Functions

- __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (const cmp_fn &r_cmp_fn)
- template<typename It >
  __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (It first_it, It last_it)

- template<typename It >
  __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (It first_it, It last_it, const cmp_fn &r_cmp_fn)
- **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue** (const priority_queue &other)
- priority_queue & **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::operator=** (const priority_queue &other)
- void **__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::swap** (priority_queue &other)

2.59.1 Detailed Description

2.59.2 Function Documentation

2.59.2.1 template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (const cmp_fn & *r_cmp_fn*) `[inline]`

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

Definition at line 116 of file priority_queue.hpp.

2.59.2.2 template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (It *first_it,* It *last_it*) `[inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 122 of file priority_queue.hpp.

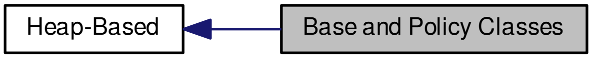
2.59.2.3 template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (It *first_it,* It *last_it,* const cmp_fn & *r_cmp_fn*) `[inline]`

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.

Definition at line 130 of file priority_queue.hpp.

2.60 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



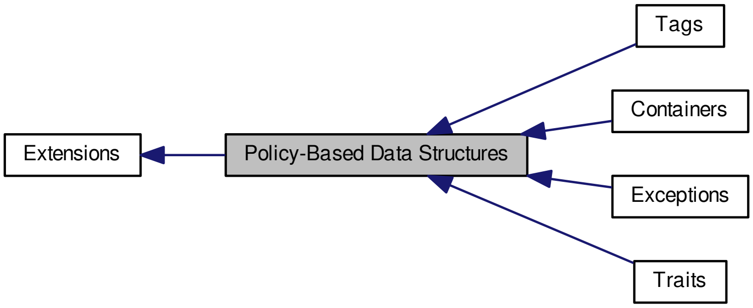
Classes

- class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

2.60.1 Detailed Description

2.61 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Modules

- Containers
- Exceptions
- Tags
- Traits

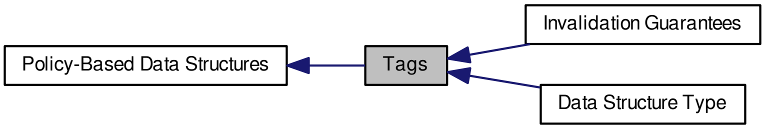
2.61.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: `http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html`

2.62 Tags

Collaboration diagram for Tags:



Modules

- Data Structure Type
- Invalidation Guarantees

Classes

- struct __gnu_pbds::trivial_iterator_tag

  A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.

Typedefs

- typedef void __gnu_pbds::trivial_iterator_difference_type

2.62.1 Detailed Description

2.62.2 Typedef Documentation

2.62.2.1 typedef void __gnu_pbds::trivial_iterator_difference_type

Prohibit moving trivial iterators.

Definition at line 79 of file tag_and_trait.hpp.

2.63 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:

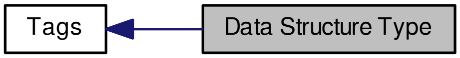
Classes

- struct __gnu_pbds::basic_invalidation_guarantee
- struct __gnu_pbds::point_invalidation_guarantee
- struct __gnu_pbds::range_invalidation_guarantee

2.63.1 Detailed Description

2.64 Data Structure Type

Collaboration diagram for Data Structure Type:



Classes

- struct __gnu_pbds::associative_tag

    Basic associative-container.
- struct __gnu_pbds::basic_branch_tag

    Basic branch structure.
- struct __gnu_pbds::basic_hash_tag

    Basic hash structure.
- struct __gnu_pbds::binary_heap_tag

    Binary-heap (array-based).
- struct __gnu_pbds::binomial_heap_tag

    Binomial-heap.
- struct __gnu_pbds::cc_hash_tag

    Collision-chaining hash.
- struct __gnu_pbds::container_tag

    Base data structure tag.
- struct __gnu_pbds::gp_hash_tag

    General-probing hash.
- struct __gnu_pbds::list_update_tag

    List-update.
- struct __gnu_pbds::ov_tree_tag

    Ordered-vector tree.
- struct __gnu_pbds::pairing_heap_tag

    Pairing-heap.
- struct __gnu_pbds::pat_trie_tag

    PATRICIA trie.
- struct __gnu_pbds::priority_queue_tag

    Basic priority-queue.
- struct __gnu_pbds::rb_tree_tag

    Red-black tree.
- struct __gnu_pbds::rc_binomial_heap_tag

    Redundant-counter binomial-heap.
- struct __gnu_pbds::sequence_tag

    Basic sequence.

- struct __gnu_pbds::splay_tree_tag

  Splay tree.
- struct __gnu_pbds::string_tag

  Basic string container, inclusive of strings, ropes, etc.
- struct __gnu_pbds::thin_heap_tag

  Thin heap.
- struct __gnu_pbds::tree_tag

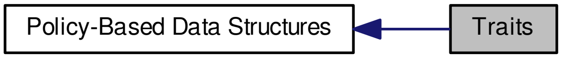
  Basic tree structure.
- struct __gnu_pbds::trie_tag

  Basic trie structure.

2.64.1 Detailed Description

2.65 Traits

Collaboration diagram for Traits:



Classes

- struct __gnu_pbds::container_traits< Cntnr >

  Container traits.
- struct __gnu_pbds::container_traits_base< binary_heap_tag >

  Specialization, binary heap.
- struct __gnu_pbds::container_traits_base< binomial_heap_tag >

  Specialization, binomial heap.
- struct __gnu_pbds::container_traits_base< cc_hash_tag >

  Specialization, cc hash.
- struct __gnu_pbds::container_traits_base< gp_hash_tag >

  Specialization, gp hash.
- struct __gnu_pbds::container_traits_base< list_update_tag >

  Specialization, list update.
- struct __gnu_pbds::container_traits_base< ov_tree_tag >

  Specialization, ov tree.
- struct __gnu_pbds::container_traits_base< pairing_heap_tag >

  Specialization, pairing heap.
- struct __gnu_pbds::container_traits_base< pat_trie_tag >

  Specialization, pat trie.
- struct __gnu_pbds::container_traits_base< rb_tree_tag >

  Specialization, rb tree.
- struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >

  Specialization, rc binomial heap.
- struct __gnu_pbds::container_traits_base< splay_tree_tag >

  Specialization, splay tree.
- struct __gnu_pbds::container_traits_base< thin_heap_tag >

  Specialization, thin heap.
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >

  Binary search tree traits, primary template.
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >

  Primary template.

- struct __gnu_pbds::detail::no_throw_copies< Key, null_type >

  Specialization.
- struct __gnu_pbds::detail::stored_data< _Tv, _Th >

  Primary template for representation of stored data. Two types of data can be stored: value and hash.
- struct __gnu_pbds::detail::stored_data< _Tv, null_type >

  Specialization for representation of stored data of just value type.
- struct __gnu_pbds::detail::stored_hash< _Th >

  Stored hash.
- struct __gnu_pbds::detail::stored_value< _Tv >

  Stored value.
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >

  Specialization, false.
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >

  Specialization, true.
- struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

  Tree node metadata dispatch.
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

  Tree traits.
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >

  Specialization, false.
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >

  Specialization, true.
- struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

  Trie node metadata dispatch.
- struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >
- struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >
- struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >
- struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >
- struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >

  Type base dispatch.
- struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

  Traits for abstract types.
- struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >

A null node updator, indicating that no node updates are required.

- struct __gnu_pbds::null_type

  Represents no type, or absence of type, for template tricks.
- struct container_traits_base< _Tag >

  Primary template, container traits base.
- struct tree_metadata_helper< Node_Update, _BTp >

  Tree metadata helper.
- struct trie_metadata_helper< Node_Update, _BTp >

  Trie metadata helper.
- struct type_base< Key, Mapped, _Alloc, Store_Hash >

  Primary template.

Variables

- static null_type **__gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >::s_null_type**
- static null_type **__gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >::s_null_type**

2.65.1 Detailed Description

2.66 Random Number Generators

Collaboration diagram for Random Number Generators:

Classes

- class std::discard_block_engine< _RandomNumberEngine, __p, __r >
- class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
- class std::linear_congruential_engine< _UIntType, __a, __c, __m >

  A model of a linear congruential random number generator.
- class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
- class std::random_device
- class std::shuffle_order_engine< _RandomNumberEngine, __k >

  Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine
  < minstd_rand0, 256 > **std::knuth_b**
- typedef
  linear_congruential_engine
  < uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand
- typedef
  linear_congruential_engine
  < uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0
- typedef
  mersenne_twister_engine
  < uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937
- typedef
  mersenne_twister_engine
  < uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64
- typedef discard_block_engine
  < ranlux24_base, 223, 23 > **std::ranlux24**
- typedef
  subtract_with_carry_engine
  < uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**

- typedef discard_block_engine
  < ranlux48_base, 389, 11 > **std::ranlux48**
- typedef
  subtract_with_carry_engine
  < uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m>
  bool std::operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
  bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)
- template<typename _UIntType , size_t __w, size_t __s, size_t __r>
  bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)
- template<typename _RandomNumberEngine , size_t __p, size_t __r>
  bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)
- template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
  bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)
- template<typename _RandomNumberEngine , size_t __k>
  bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)
- template<typename _RandomNumberEngine , size_t __w, typename _UIntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

2.66.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an operator() that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

To be documented.

Table 273: Random Number Generator Requirements

2.66.2 Typedef Documentation

2.66.2.1 typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand

An alternative LCR (Lehmer Generator function).

Definition at line 1527 of file random.h.

2.66.2.2 typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

Definition at line 1521 of file random.h.

2.66.2.3 typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo--Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1543 of file random.h.

2.66.2.4 typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL> std::mt19937_64

An alternative Mersenne Twister.

Definition at line 1555 of file random.h.

2.66.3 Function Documentation

2.66.3.1 template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m> bool std::operator!= (const **std::linear_congruential_engine**< _UIntType, __a, __c, __m > & *__lhs*, const **std::linear_congruential_engine**< _UIntType, __a, __c, __m > & *__rhs*) `[inline]`

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

__lhs	A linear congruential random number generator object.
__rhs	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 409 of file random.h.

2.66.3.2 template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool std::operator!= (const **std::mersenne_twister_engine**< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & *__lhs*, const **std::mersenne_twister_engine**< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & *__rhs*) `[inline]`

Compares two % mersenne_twister_engine random number generator objects of the same type for inequality.

Parameters

__lhs	A % mersenne_twister_engine random number generator object.
__rhs	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 641 of file random.h.

2.66.3.3 template<typename _UIntType , size_t __w, size_t __s, size_t __r> bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & *__lhs,* const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & *__rhs*) [inline]

Compares two % subtract_with_carry_engine random number generator objects of the same type for inequality.

Parameters

__lhs	A % subtract_with_carry_engine random number generator object.
__rhs	Another % subtract_with_carry_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 840 of file random.h.

2.66.3.4 template<typename _RandomNumberEngine , size_t __p, size_t __r> bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > & *__lhs,* const std::discard_block_engine< _RandomNumberEngine, __p, __r > & *__rhs*) [inline]

Compares two discard_block_engine random number generator objects of the same type for inequality.

Parameters

__lhs	A discard_block_engine random number generator object.
__rhs	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1062 of file random.h.

2.66.3.5 template<typename _RandomNumberEngine , size_t __w, typename _UIntType > bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & *__lhs,* const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & *__rhs*) [inline]

Compares two independent_bits_engine random number generator objects of the same type for inequality.

Parameters

__lhs	A independent_bits_engine random number generator object.
__rhs	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1258 of file random.h.

2.66.3.6 template<typename _RandomNumberEngine , size_t __k> bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > & *__lhs*, const std::shuffle_order_engine< _RandomNumberEngine, __k > & *__rhs*) [inline]

Compares two shuffle_order_engine random number generator objects of the same type for inequality.

Parameters

__lhs	A shuffle_order_engine random number generator object.
__rhs	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1510 of file random.h.

2.66.3.7 template<typename _RandomNumberEngine , size_t __w, typename _UIntType , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& std::operator<< (std::basic_ostream< _CharT, _Traits > & *__os*, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & *__x*)

Inserts the current state of a independent_bits_engine random number generator engine `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A independent_bits_engine random number generator engine.

Returns

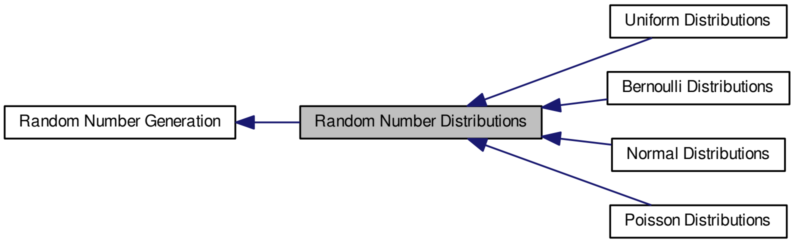
The output stream with the state of `__x` inserted or in an error state.

Definition at line 1277 of file random.h.

References std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::base().

2.67 Random Number Distributions

Collaboration diagram for Random Number Distributions:



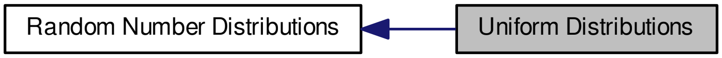
Modules

- Bernoulli Distributions
- Normal Distributions
- Poisson Distributions
- Uniform Distributions

2.67.1 Detailed Description

2.68 Uniform Distributions

Collaboration diagram for Uniform Distributions:



Classes

- class std::uniform_int_distribution< _IntType >

  Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ *with equal probability throughout the range.*
- struct std::uniform_int_distribution< _IntType >::param_type
- class std::uniform_real_distribution< _RealType >

  Uniform continuous distribution for random numbers.
- struct std::uniform_real_distribution< _RealType >::param_type

Functions

- template<typename _IntType >
  bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)

2.68.1 Detailed Description

2.68.2 Function Documentation

2.68.2.1 template< typename _IntType > bool std::operator!= (const **std::uniform_int_distribution**< _IntType > & *__d1*, const **std::uniform_int_distribution**< _IntType > & *__d2*) `[inline]`

Return true if two uniform integer distributions have different parameters.

Definition at line 1825 of file random.h.

2.68.2.2 template< typename _IntType > bool std::operator!= (const **std::uniform_real_distribution**< _IntType > & *__d1*, const **std::uniform_real_distribution**< _IntType > & *__d2*) `[inline]`

Return true if two uniform real distributions have different parameters.

Definition at line 2034 of file random.h.

2.68.2.3 template< typename _IntType , typename _CharT , typename _Traits > **std::basic_ostream**< _CharT, _Traits > & std::operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::uniform_int_distribution**< _IntType > & *__x*)

Inserts a uniform_int_distribution random number distribution ___`x` into the output stream `os`.

Parameters

__os	An output stream.
__x	A uniform_int_distribution random number distribution.

Returns

The output stream with the state of ___`x` inserted or in an error state.

Definition at line 1029 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.68.2.4 template< typename _RealType , typename _CharT , typename _Traits > **std::basic_ostream**< _CharT, _Traits > & std::operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::uniform_real_distribution**< _RealType > & *__x*)

Inserts a uniform_real_distribution random number distribution ___`x` into the output stream ___`os`.

Parameters

__os	An output stream.
__x	A uniform_real_distribution random number distribution.

Returns

The output stream with the state of ___`x` inserted or in an error state.

Definition at line 1088 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.68.2.5 template<typename _IntType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & *__is,* std::uniform_int_distribution< _IntType > & *__x*)

Extracts a uniform_int_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A uniform_int_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1050 of file bits/random.tcc.

References std::dec(), std::ios_base::flags(), std::uniform_int_distribution< _IntType >::param(), and std::skipws().

2.68.2.6 template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & *__is,* std::uniform_real_distribution< _RealType > & *__x*)

Extracts a uniform_real_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A uniform_real_distribution random number generator engine.

Returns

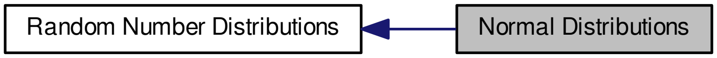
The input stream with `__x` extracted or in an error state.

Definition at line 1112 of file bits/random.tcc.

References std::ios_base::flags(), std::uniform_real_distribution< _RealType >::param(), and std::skipws().

2.69 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class std::cauchy_distribution< _RealType >

    A cauchy_distribution random number distribution.
- struct std::cauchy_distribution< _RealType >::param_type
- class std::chi_squared_distribution< _RealType >

    A chi_squared_distribution random number distribution.
- struct std::chi_squared_distribution< _RealType >::param_type
- class std::fisher_f_distribution< _RealType >

    A fisher_f_distribution random number distribution.
- struct std::fisher_f_distribution< _RealType >::param_type
- class std::gamma_distribution< _RealType >

    A gamma continuous distribution for random numbers.
- struct std::gamma_distribution< _RealType >::param_type
- class std::lognormal_distribution< _RealType >

    A lognormal_distribution random number distribution.
- struct std::lognormal_distribution< _RealType >::param_type
- class std::normal_distribution< _RealType >

    A normal continuous distribution for random numbers.
- struct std::normal_distribution< _RealType >::param_type
- class std::student_t_distribution< _RealType >

    A student_t_distribution random number distribution.
- struct std::student_t_distribution< _RealType >::param_type

Functions

- template<typename _RealType >
  bool std::operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)

- template<typename _RealType >
  bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)

2.69.1 Detailed Description

2.69.2 Function Documentation

2.69.2.1 template<typename _RealType > bool std::operator!= (const **std::normal_distribution**< _RealType > & *__d1,* const **std::normal_distribution**< _RealType > & *__d2*) [inline]

Return true if two normal distributions are different.

Definition at line 2283 of file random.h.

2.69.2.2 template<typename _RealType > bool std::operator!= (const **std::lognormal_distribution**< _RealType > & *__d1,* const **std::lognormal_distribution**< _RealType > & *__d2*) [inline]

Return true if two lognormal distributions are different.

Definition at line 2487 of file random.h.

2.69.2.3 template<typename _RealType > bool std::operator!= (const **std::gamma_distribution**< _RealType > & *__d1,* const **std::gamma_distribution**< _RealType > & *__d2*) [inline]

Return true if two gamma distributions are different.

Definition at line 2707 of file random.h.

2.69.2.4 template<typename _RealType > bool std::operator!= (const **std::chi_squared_distribution**< _RealType > & *__d1,* const **std::chi_squared_distribution**< _RealType > & *__d2*) [inline]

Return true if two Chi-squared distributions are different.

Definition at line 2917 of file random.h.

2.69.2.5 template<typename _RealType > bool std::operator!= (const **std::cauchy_distribution**< _RealType > & *__d1,* const **std::cauchy_distribution**< _RealType > & *__d2*) [inline]

Return true if two Cauchy distributions have different parameters.

Definition at line 3084 of file random.h.

2.69.2.6 template< typename _RealType > bool std::operator!= (const **std::fisher_f_distribution**< _RealType > & *__d1*, const **std::fisher_f_distribution**< _RealType > & *__d2*) [inline]

Return true if two Fisher f distributions are diferent.

Definition at line 3340 of file random.h.

2.69.2.7 template< typename _RealType > bool std::operator!= (const **std::student_t_distribution**< _RealType > & *__d1*, const **std::student_t_distribution**< _RealType > & *__d2*) [inline]

Return true if two Student t distributions are different.

Definition at line 3553 of file random.h.

2.69.2.8 template< typename _RealType , typename _CharT , typename _Traits > **std::basic_ostream**< _CharT, _Traits > & std::operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::cauchy_distribution**< _RealType > & *__x*)

Inserts a cauchy_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A cauchy_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2273 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.69.2.9 template< typename _RealType , typename _CharT , typename _Traits > **std::basic_istream**< _CharT, _Traits > & std::operator>> (**std::basic_istream**< _CharT, _Traits > & *__is*, **std::cauchy_distribution**< _RealType > & *__x*)

Extracts a cauchy_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A cauchy_distribution random number generator engine.

Returns

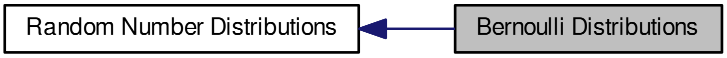
The input stream with `__x` extracted or in an error state.

Definition at line 2297 of file bits/random.tcc.

References std::dec(), std::ios_base::flags(), std::cauchy_distribution< _RealType >::param(), and std::skipws().

2.70 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class std::bernoulli_distribution

    A Bernoulli random number distribution.
- struct std::bernoulli_distribution::param_type
- class std::binomial_distribution< _IntType >

    A discrete binomial random number distribution.
- struct std::binomial_distribution< _IntType >::param_type
- class std::geometric_distribution< _IntType >

    A discrete geometric random number distribution.
- struct std::geometric_distribution< _IntType >::param_type
- class std::negative_binomial_distribution< _IntType >

    A negative_binomial_distribution random number distribution.
- struct std::negative_binomial_distribution< _IntType >::param_type

Functions

- bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)
- template<typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)

- template<typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)

2.70.1 Detailed Description

2.70.2 Function Documentation

2.70.2.1 bool std::operator!= (const **std::bernoulli_distribution** & *__d1*, const **std::bernoulli_distribution** & *__d2*) `[inline]`

Return true if two Bernoulli distributions have different parameters.

Definition at line 3729 of file random.h.

2.70.2.2 template<typename _IntType > bool std::operator!= (const **std::binomial_distribution**< _IntType > & *__d1*, const **std::binomial_distribution**< _IntType > & *__d2*) `[inline]`

Return true if two binomial distributions are different.

Definition at line 3994 of file random.h.

2.70.2.3 template<typename _IntType > bool std::operator!= (const **std::geometric_distribution**< _IntType > & *__d1*, const **std::geometric_distribution**< _IntType > & *__d2*) `[inline]`

Return true if two geometric distributions have different parameters.

Definition at line 4163 of file random.h.

2.70.2.4 template<typename _IntType > bool std::operator!= (const **std::negative_binomial_distribution**< _IntType > & *__d1*, const **std::negative_binomial_distribution**< _IntType > & *__d2*) `[inline]`

Return true if two negative binomial distributions are different.

Definition at line 4408 of file random.h.

2.70.2.5 template<typename _CharT , typename _Traits > **std::basic_ostream**< _CharT, _Traits > & std::operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::bernoulli_distribution** & *__x*)

Inserts a bernoulli_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A bernoulli_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1150 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.70.2.6 template<typename _IntType , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::geometric_distribution< _IntType > & __x)

Inserts a geometric_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A geometric_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1232 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.70.2.7 template<typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::bernoulli_distribution & __x)

Extracts a bernoulli_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A bernoulli_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 3759 of file random.h.

References std::bernoulli_distribution::param().

2.70.2.8 template<typename _IntType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & __is, std::geometric_distribution< _IntType > & __x)

Extracts a geometric_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A geometric_distribution random number generator engine.

Returns

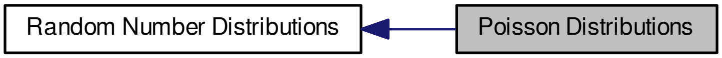
The input stream with `__x` extracted or in an error state.

Definition at line 1256 of file bits/random.tcc.

References std::ios_base::flags(), std::geometric_distribution< _IntType >::param(), and std::skipws().

2.71 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class std::discrete_distribution< _IntType >

  A discrete_distribution random number distribution.
- struct std::discrete_distribution< _IntType >::param_type
- class std::exponential_distribution< _RealType >

  An exponential continuous distribution for random numbers.
- struct std::exponential_distribution< _RealType >::param_type
- class std::extreme_value_distribution< _RealType >

  A extreme_value_distribution random number distribution.
- struct std::extreme_value_distribution< _RealType >::param_type
- class std::piecewise_constant_distribution< _RealType >

  A piecewise_constant_distribution random number distribution.
- struct std::piecewise_constant_distribution< _RealType >::param_type
- class std::piecewise_linear_distribution< _RealType >

  A piecewise_linear_distribution random number distribution.
- struct std::piecewise_linear_distribution< _RealType >::param_type
- class std::poisson_distribution< _IntType >

  A discrete Poisson random number distribution.
- struct std::poisson_distribution< _IntType >::param_type
- class std::weibull_distribution< _RealType >

  A weibull_distribution random number distribution.
- struct std::weibull_distribution< _RealType >::param_type

Functions

- template<typename _IntType >
  bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)

- template<typename _RealType >
  bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)

2.71.1 Detailed Description

2.71.2 Function Documentation

2.71.2.1 template<typename _IntType > bool std::operator!= (const **std::poisson_distribution**< _IntType > & *__d1,* const **std::poisson_distribution**< _IntType > & *__d2*) [inline]

Return true if two Poisson distributions are different.

Definition at line 4624 of file random.h.

2.71.2.2 template<typename _RealType > bool std::operator!= (const **std::exponential_distribution**< _RealType > & *__d1,* const **std::exponential_distribution**< _RealType > & *__d2*) [inline]

Return true if two exponential distributions have different parameters.

Definition at line 4802 of file random.h.

2.71.2.3 template<typename _RealType > bool std::operator!= (const std::weibull_distribution< _RealType > & __d1, const std::weibull_distribution< _RealType > & __d2) [inline]

Return true if two Weibull distributions have different parameters.

Definition at line 5005 of file random.h.

2.71.2.4 template<typename _RealType > bool std::operator!= (const std::extreme_value_distribution< _RealType > & __d1, const std::extreme_value_distribution< _RealType > & __d2) [inline]

Return true if two extreme value distributions have different parameters.

Definition at line 5208 of file random.h.

2.71.2.5 template<typename _IntType > bool std::operator!= (const std::discrete_distribution< _IntType > & __d1, const std::discrete_distribution< _IntType > & __d2) [inline]

Return true if two discrete distributions have different parameters.

Definition at line 5468 of file random.h.

2.71.2.6 template<typename _RealType > bool std::operator!= (const std::piecewise_constant_distribution< _RealType > & __d1, const std::piecewise_constant_distribution< _RealType > & __d2) [inline]

Return true if two piecewise constant distributions have different parameters.

Definition at line 5735 of file random.h.

2.71.2.7 template<typename _RealType > bool std::operator!= (const std::piecewise_linear_distribution< _RealType > & __d1, const std::piecewise_linear_distribution< _RealType > & __d2) [inline]

Return true if two piecewise linear distributions have different parameters.

Definition at line 6005 of file random.h.

2.71.2.8 template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::exponential_distribution< _RealType > & __x)

Inserts a exponential_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A exponential_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1881 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.71.2.9 template<typename _RealType , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::weibull_distribution< _RealType > & __x)

Inserts a weibull_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A weibull_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2672 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.71.2.10 template< typename _RealType , typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > & *__os,* const std::extreme_value_distribution< _RealType > & *__x*)

Inserts a extreme_value_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A extreme_value_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2748 of file bits/random.tcc.

References std::ios_base::flags(), std::left(), and std::scientific().

2.71.2.11 template< typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & *__is,* std::exponential_distribution< _RealType > & *__x*)

Extracts a exponential_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A exponential_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1904 of file bits/random.tcc.

References std::dec(), std::ios_base::flags(), std::exponential_distribution< _RealType >::param(), and std::skipws().

2.71.2.12 template< typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & *__is,* std::weibull_distribution< _RealType > & *__x*)

Extracts a weibull_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A weibull_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2696 of file bits/random.tcc.

References std::dec(), std::ios_base::flags(), std::weibull_distribution< _RealType >::param(), and std::skipws().

2.71.2.13 template<typename _RealType , typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > & *__is,* std::extreme_value_distribution< _RealType > & *__x*)

Extracts a extreme_value_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A extreme_value_distribution random number generator engine.

Returns

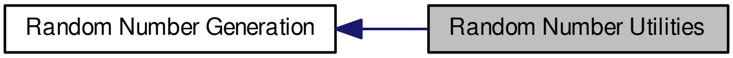
The input stream with `__x` extracted or in an error state.

Definition at line 2772 of file bits/random.tcc.

References std::dec(), std::ios_base::flags(), std::extreme_value_distribution< _RealType >::param(), and std::skipws().

2.72 Random Number Utilities

Collaboration diagram for Random Number Utilities:



Classes

- class std::seed_seq

  The seed_seq class generates sequences of seeds for random number generators.

2.72.1 Detailed Description

3 Namespace Documentation

3.1 __gnu_cxx Namespace Reference

Namespaces

- namespace __detail
- namespace typelist

Classes

- struct __alloc_traits

  Uniform interface to C++98 and C++0x allocators.
- struct __common_pool_policy

  Policy for shared __pool objects.
- class __mt_alloc

  This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html`.
- class __mt_alloc_base

  Base class for _Tp dependent member functions.
- struct __per_type_pool_policy

  Policy for individual __pool objects.
- class __pool< false >

  Specialization for single thread.
- class __pool< true >

  Specialization for thread enabled, via gthreads.h.
- class __pool_alloc

  Allocator using a memory pool with a single lock.
- class __pool_alloc_base

  Base class for __pool_alloc.
- struct __pool_base

  Base class for pool object.
- class __rc_string_base
- class __scoped_lock

  Scoped lock idiom.
- class __versa_string

  Template class __versa_string.
  Data structure managing sequences of characters and character-like objects.
- struct _Caster
- struct _Char_types

  Mapping from character type to associated types.
- class _ExtPtr_allocator

  An example allocator which uses a non-standard pointer type.
  This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.
- struct _Invalid_type
- class _Pointer_adapter
- class _Relative_pointer_impl

A storage policy for use with _Pointer_adapter<> which stores the pointer's address as an offset value which is relative to its own address.

- class _Relative_pointer_impl< const _Tp >
- class _Std_pointer_impl

  A storage policy for use with _Pointer_adapter<> which yields a standard pointer.
- struct _Unqualified_type
- struct annotate_base

  Base class for checking address and label information about allocations. Create a std::map between the allocated address (void∗) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.
- class array_allocator

  An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class array_allocator_base

  Base class.
- class binary_compose

  An SGI extension .
- class bitmap_allocator

  Bitmap Allocator, primary template.
- struct char_traits

  Base class used to implement std::char_traits.
- struct character

  A POD class that serves as a character abstraction class.
- struct condition_base

  Base struct for condition policy.
- struct constant_binary_fun

  An SGI extension .
- struct constant_unary_fun

  An SGI extension .
- struct constant_void_fun

  An SGI extension .
- class debug_allocator

  A meta-allocator with debugging bits, as per [20.4].
  This is precisely the allocator defined in the C++ Standard.
- class enc_filebuf

  class enc_filebuf.
- struct encoding_char_traits

  encoding_char_traits
- class encoding_state

  Extension to use iconv for dealing with character encodings.
- struct forced_error

  Thown by exception safety machinery.
- class free_list

  The free list class for managing chunks of memory to be given to and returned by the bitmap_allocator.
- class hash_map
- class hash_multimap
- class hash_multiset
- class hash_set
- struct limit_condition

  Base class for incremental control and throw.

- class malloc_allocator

  An allocator that uses malloc.
  This is precisely the allocator defined in the C++ Standard.
- class new_allocator

  An allocator that uses global new, as per [20.4].
  This is precisely the allocator defined in the C++ Standard.
- struct project1st

  An SGI extension .
- struct project2nd

  An SGI extension .
- struct random_condition

  Base class for random probability control and throw.
- struct rb_tree
- class recursive_init_error

  Exception thrown by __cxa_guard_acquire.
  6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.
- class rope
- struct select1st

  An SGI extension .
- struct select2nd

  An SGI extension .
- class slist
- class stdio_filebuf

  Provides a layer of compatibility for C/POSIX.
  This GNU extension provides extensions for working with standard C FILE∗'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.
- class stdio_sync_filebuf

  Provides a layer of compatibility for C.
  This GNU extension provides extensions for working with standard C FILE∗'s. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.
- class subtractive_rng
- struct temporary_buffer
- class throw_allocator_base

  Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templatized code.
  Note: Deallocate not allowed to throw.
- struct throw_allocator_limit

  Allocator throwing via limit condition.
- struct throw_allocator_random

  Allocator throwing via random condition.
- struct throw_value_base

  Class with exception generation control. Intended to be used as a value_type in templatized code.
- struct throw_value_limit

  Type throwing via limit condition.
- struct throw_value_random

  Type throwing via random condition.
- class unary_compose

  An SGI extension .

Typedefs

- typedef void(∗ **__destroy_handler**)(void ∗)
- typedef __versa_string< char,
  std::char_traits< char >
  , std::allocator< char >
  , __rc_string_base > **__rc_string**
- typedef __vstring **__sso_string**
- typedef __versa_string
  < char16_t, std::char_traits
  < char16_t >, std::allocator
  < char16_t >, __rc_string_base > **__u16rc_string**
- typedef __u16vstring **__u16sso_string**
- typedef __versa_string< char16_t > **__u16vstring**
- typedef __versa_string
  < char32_t, std::char_traits
  < char32_t >, std::allocator
  < char32_t >, __rc_string_base > **__u32rc_string**
- typedef __u32vstring **__u32sso_string**
- typedef __versa_string< char32_t > **__u32vstring**
- typedef __versa_string< char > **__vstring**
- typedef __versa_string
  < wchar_t, std::char_traits
  < wchar_t >, std::allocator
  < wchar_t >, __rc_string_base > **__wrc_string**
- typedef __wvstring **__wsso_string**
- typedef __versa_string< wchar_t > **__wvstring**
- typedef rope< char > **crope**
- typedef rope< wchar_t > **wrope**

Enumerations

- enum { **_S_num_primes** }
- enum **_Lock_policy** { **_S_single**, **_S_mutex**, **_S_atomic** }

Functions

- static void **__atomic_add_single** (_Atomic_word ∗__mem, int __val)
- else **__atomic_add_single** (__mem, __val)
- _Atomic_word **__attribute__** ((__unused__)) __exchange_and_add(volatile _Atomic_word ∗
- template<class _Tp >
  void **__aux_require_boolean_expr** (const _Tp &__t)
- template<typename _ToType , typename _FromType >
  _ToType **__const_pointer_cast** (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__const_pointer_cast** (_FromType ∗__arg)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  pair< _InputIterator,
  _OutputIterator > **__copy_n** (_InputIterator __first, _Size __count, _OutputIterator __result, input_iterator_tag)

- template<typename _RAIterator , typename _Size , typename _OutputIterator >
  pair< _RAIterator,
  _OutputIterator > **__copy_n** (_RAIterator __first, _Size __count, _OutputIterator __result, random_access_iterator_tag)
- template<typename _InputIterator , typename _Distance >
  void **__distance** (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Distance >
  void **__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag)
- template<typename _ToType , typename _FromType >
  _ToType **__dynamic_pointer_cast** (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__dynamic_pointer_cast** (_FromType ∗__arg)
- void **__error_type_must_be_a_signed_integer_type** ()
- void **__error_type_must_be_an_integer_type** ()
- void **__error_type_must_be_an_unsigned_integer_type** ()
- static _Atomic_word **__exchange_and_add_single** (_Atomic_word ∗__mem, int __val)
- else return **__exchange_and_add_single** (__mem, __val)
- template<class _Concept >
  void **__function_requires** ()
- template<typename _Type >
  bool **__is_null_pointer** (_Type ∗__ptr)
- template<typename _Type >
  bool **__is_null_pointer** (_Type)
- template<typename _InputIterator1 , typename _InputIterator2 >
  int **__lexicographical_compare_3way** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- int **__lexicographical_compare_3way** (const unsigned char ∗__first1, const unsigned char ∗__last1, const unsigned char ∗__first2, const unsigned char ∗__last2)
- int **__lexicographical_compare_3way** (const char ∗__first1, const char ∗__last1, const char ∗__first2, const char ∗__last2)
- template<typename _Tp >
  const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp , typename _Compare >
  const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- crope::reference **__mutable_reference_at** (crope &__c, size_t __i)
- template<typename _Tp , typename _Integer , typename _MonoidOperation >
  _Tp **__power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _Tp , typename _Integer >
  _Tp **__power** (_Tp __x, _Integer __n)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _Distance >
  _RandomAccessIterator **__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator , typename _Distance >
  _RandomAccessIterator **__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- template<typename _ToType , typename _FromType >
  _ToType **__reinterpret_pointer_cast** (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__reinterpret_pointer_cast** (_FromType ∗__arg)
- _Slist_node_base ∗ **__slist_make_link** (_Slist_node_base ∗__prev_node, _Slist_node_base ∗__new_node)
- _Slist_node_base ∗ **__slist_previous** (_Slist_node_base ∗__head, const _Slist_node_base ∗__node)

- const _Slist_node_base ∗ **__slist_previous** (const _Slist_node_base ∗__head, const _Slist_node_base ∗__node)
- _Slist_node_base ∗ **__slist_reverse** (_Slist_node_base ∗__node)
- size_t **__slist_size** (_Slist_node_base ∗__node)
- void **__slist_splice_after** (_Slist_node_base ∗__pos, _Slist_node_base ∗__before_first, _Slist_node_base ∗__before_last)
- void **__slist_splice_after** (_Slist_node_base ∗__pos, _Slist_node_base ∗__head)
- template<typename _ToType , typename _FromType >
  _ToType __static_pointer_cast (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType __static_pointer_cast (_FromType ∗__arg)
- size_t **__stl_hash_string** (const char ∗__s)
- unsigned long **__stl_next_prime** (unsigned long __n)
- template<typename _TRet , typename _Ret = _TRet, typename _CharT , typename... _Base>
  _Ret **__stoa** (_TRet(∗__convf)(const _CharT ∗, _CharT ∗∗, _Base...), const char ∗__name, const _CharT ∗__str, std::size_t ∗__idx, _Base...__base)
- void **__throw_concurrence_lock_error** ()
- void **__throw_concurrence_unlock_error** ()
- void **__throw_forced_error** ()
- template<typename _String , typename _CharT = typename _String::value_type>
  _String **__to_xstring** (int(∗__convf)(_CharT ∗, std::size_t, const _CharT ∗, __builtin_va_list), std::size_t __n, const _CharT ∗__fmt,...)
- template<typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > **__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)
- template<typename _RandomAccessIter , typename _Size , typename _ForwardIter >
  pair< _RandomAccessIter,
  _ForwardIter > **__uninitialized_copy_n** (_RandomAccessIter __first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)
- template<typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > **__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result)
- template<typename _InputIter , typename _Size , typename _ForwardIter , typename _Allocator >
  pair< _InputIter, _ForwardIter > **__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)
- template<typename _InputIter , typename _Size , typename _ForwardIter , typename _Tp >
  pair< _InputIter, _ForwardIter > **__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)
- void __verbose_terminate_handler ()
- size_t _Bit_scan_forward (size_t __num)
- template<typename _ForwardIterator , typename _Allocator >
  void **_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)
- template<typename _ForwardIterator , typename _Tp >
  void **_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp >)
- template<class _CharT , class _Traits >
  void **_Rope_fill** (basic_ostream< _CharT, _Traits > &__o, size_t __n)
- template<class _CharT >
  bool **_Rope_is_simple** (_CharT ∗)
- bool **_Rope_is_simple** (char ∗)
- bool **_Rope_is_simple** (wchar_t ∗)
- template<class _Rope_iterator >
  void **_Rope_rotate** (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)

- template<class _CharT >
  void **_S_cond_store_eos** (_CharT &)
- void **_S_cond_store_eos** (char &__c)
- void **_S_cond_store_eos** (wchar_t &__c)
- template<class _CharT >
  _CharT **_S_eos** (_CharT ∗)
- template<class _CharT >
  bool **_S_is_basic_char_type** (_CharT ∗)
- bool **_S_is_basic_char_type** (char ∗)
- bool **_S_is_basic_char_type** (wchar_t ∗)
- template<class _CharT >
  bool **_S_is_one_byte_char_type** (_CharT ∗)
- bool **_S_is_one_byte_char_type** (char ∗)
- template<class _Operation1 , class _Operation2 >
  unary_compose< _Operation1,
  _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1 , class _Operation2 , class _Operation3 >
  binary_compose< _Operation1,
  _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result >
  constant_void_fun< _Result > constant0 (const _Result &__val)
- template<class _Result >
  constant_unary_fun< _Result,
  _Result > constant1 (const _Result &__val)
- template<class _Result >
  constant_binary_fun< _Result,
  _Result, _Result > constant2 (const _Result &__val)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  pair< _InputIterator,
  _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)
- template<typename _InputIterator , typename _Tp , typename _Size >
  void **count** (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- template<typename _InputIterator , typename _Predicate , typename _Size >
  void **count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- template<typename _InputIterator , typename _Distance >
  void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<class _Tp >
  _Tp identity_element (std::plus< _Tp >)
- template<class _Tp >
  _Tp identity_element (std::multiplies< _Tp >)
- static _Atomic_word int __val **if** (__gthread_active_p()) return __exchange_and_add(__mem
- template<typename _InputIterator1 , typename _InputIterator2 >
  int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<class _Ret , class _Tp , class _Arg >
  mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun1** (_Ret(_Tp::∗__f)(_Arg))
- template<class _Ret , class _Tp , class _Arg >
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun1_ref** (_Ret(_Tp::∗__f)(_Arg))
- template<typename _Tp >
  bool **operator!=** (const new_allocator< _Tp > &, const new_allocator< _Tp > &)
- template<typename _Tp >
  bool **operator!=** (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)

- template<typename _Tp , typename _Array >
  bool **operator!=** (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)
- template<typename _Tp >
  bool **operator!=** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)
- template<class _Value , class _HashFcn , class _EqualKey , class _Alloc >
  bool **operator!=** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  bool **operator!=** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  bool **operator!=** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Key , class _Tp , class _HF , class _EqKey , class _Alloc >
  bool **operator!=** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator!=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **operator!=** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp >
  bool **operator!=** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp >
  bool **operator!=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp , typename _Cond >
  bool **operator!=** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<class _Val , class _Key , class _HF , class _Ex , class _Eq , class _All >
  bool **operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<typename _Tp , typename _Poolp >
  bool **operator!=** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)
- template<class _Tp , class _Alloc >
  bool **operator!=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **operator!=** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **operator!=** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator!=** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)

- template<class _CharT , class _Alloc >
  bool **operator!=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator!=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator!=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator!=** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<typename _Cond >
  throw_value_base< _Cond > **operator∗** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<class _CharT , class _Alloc >
  _Rope_const_iterator< _CharT,
  _Alloc > **operator+** (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
  _Rope_const_iterator< _CharT,
  _Alloc > **operator+** (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)
- template<class _CharT , class _Alloc >
  _Rope_iterator< _CharT, _Alloc > **operator+** (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
  _Rope_iterator< _CharT, _Alloc > **operator+** (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, const _CharT ∗__right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, _CharT __right)
- template<typename _Cond >
  throw_value_base< _Cond > **operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Iterator , typename _Container >
  __normal_iterator< _Iterator,
  _Container > **operator+** (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > operator+ (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (const _CharT ∗__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
__versa_string< _CharT,
_Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)
- template<class _CharT , class _Alloc >
rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT , class _Alloc >
rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, const _CharT ∗__right)
- template<class _CharT , class _Alloc >
rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT , class _Alloc >
_Rope_const_iterator< _CharT,
_Alloc > **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
ptrdiff_t **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
_Rope_iterator< _CharT, _Alloc > **operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
ptrdiff_t **operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<typename _Cond >
throw_value_base< _Cond > **operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
auto **operator-** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) -> decltype(__lhs.base()-__rhs.base())

- template<typename _Iterator , typename _Container >
  __normal_iterator< _Iterator,
  _Container >::difference_type **operator-** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename V , typename I , typename S >
  bool **operator<** (const character< V, I, S > &lhs, const character< V, I, S > &rhs)
- template<class _CharT , class _Alloc >
  bool **operator<** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator<** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator<** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator<** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator<** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Cond >
  bool **operator<** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<class _Tp , class _Alloc >
  bool **operator<** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **operator<** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **operator<** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator< (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT , class _Alloc >
  bool **operator<** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- std::ostream & **operator<<** (std::ostream &os, const annotate_base &__b)
- template<typename _UIntType , size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits , typename _StoreT >
  std::basic_ostream< _CharT,
  _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)

- template<size_t _Dimen, typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator**<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **operator**<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<class _Tp , class _Alloc >
  bool **operator**<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **operator**<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **operator**<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator<= (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT , class _Alloc >
  bool **operator<=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator<=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator<=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<typename V , typename I , typename S >
  bool **operator==** (const character< V, I, S > &lhs, const character< V, I, S > &rhs)
- template<typename _Tp >
  bool **operator==** (const new_allocator< _Tp > &, const new_allocator< _Tp > &)
- template<typename _Tp >
  bool **operator==** (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)
- template<typename _Tp , typename _Array >
  bool **operator==** (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)
- template<typename _Tp >
  bool **operator==** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)
- template<class _Val , class _Key , class _HF , class _Ex , class _Eq , class _All >
  bool **operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Value , class _HashFcn , class _EqualKey , class _Alloc >
  bool **operator==** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  bool **operator==** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<typename _UIntType , size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>
  bool **operator==** (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)
- template<class _CharT , class _Alloc >
  bool **operator==** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator==** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator==** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  bool **operator==** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Key , class _Tp , class _HF , class _EqKey , class _Alloc >
  bool **operator==** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator==** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator==** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)

- template<typename _Tp1 , typename _Tp2 >
  bool **operator==** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **operator==** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Cond >
  bool **operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp >
  bool **operator==** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp >
  bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<size_t _Dimen, typename _RealType >
  bool **operator==** (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)
- template<typename _Tp , typename _Cond >
  bool **operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Tp , typename _Poolp >
  bool **operator==** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)
- template<class _Tp , class _Alloc >
  bool **operator==** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **operator==** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **operator==** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator==** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , template< typename, typename, typename > class _Base>
  __enable_if< std::__is_char
  < _CharT >::__value, bool >
  ::__type **operator==** (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<class _CharT , class _Alloc >
  bool **operator==** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **operator>** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<class _Tp , class _Alloc >
  bool **operator>** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)

- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **operator>** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **operator>** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator> (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT , class _Alloc >
  bool **operator>** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator>** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator>** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp >
  bool **operator>=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<class _Tp , class _Alloc >
  bool **operator>=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **operator>=** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **operator>=** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool operator>= (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT , class _Alloc >
  bool **operator>=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator>=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **operator>=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<typename _UIntType , size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT , typename _Traits >

std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)

- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)
- template<size_t _Dimen, typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)
- template<typename _Tp , typename _Integer , typename _MonoidOperation >
  _Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _Tp , typename _Integer >
  _Tp power (_Tp __x, _Integer __n)
- template<typename _InputIterator , typename _RandomAccessIterator >
  _RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator >
  _RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator , typename _OutputIterator , typename _Distance >
  _OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename _RandomNumberGenerator >
  _OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)
- void **rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)
- template<typename _Tp >
  void **swap** (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)

- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  void **swap** (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  void **swap** (hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  void **swap** (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<typename _Cond >
  void **swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  void **swap** (hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Val , class _Key , class _HF , class _Extract , class _EqKey , class _All >
  void **swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)
- template<class _Tp , class _Alloc >
  void **swap** (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)
- template<class _CharT , class __Alloc >
  void **swap** (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT , class _Alloc >
  void **swap** (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)
- _Atomic_word int **throw** ()
- template<typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)

Variables

- static const _Lock_policy **__default_lock_policy**
- static _Atomic_word int __val **__val**
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **identity_element** (_Rope_Concat_fn< _CharT, _Alloc >)

3.1.1 Detailed Description

GNU extensions for public use.

3.1.2 Function Documentation

3.1.2.1 template<typename _ToType , typename _FromType > _ToType __gnu_cxx::__static_pointer_cast (const _FromType & *__arg*) [inline]

Casting operations for cases where _FromType is not a standard pointer. _ToType can be a standard or non-standard pointer. Given that _FromType is not a pointer, it must have a get() method that returns the standard pointer equivalent of the address it points to, and must have an element_type typedef which names the type it points to.

Definition at line 68 of file cast.h.

3.1.2.2 template<typename _ToType , typename _FromType > _ToType __gnu_cxx::__static_pointer_cast (_FromType * __arg) [inline]

Casting operations for cases where _FromType is a standard pointer. _ToType can be a standard or non-standard pointer.

Definition at line 96 of file cast.h.

3.1.2.3 size_t __gnu_cxx::_Bit_scan_forward (size_t __num) [inline]

Generic Version of the bsf instruction.

Definition at line 513 of file bitmap_allocator.h.

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object().

3.1.2.4 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]

Test difference of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

Definition at line 2279 of file vstring.h.

3.1.2.5 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]

Test difference of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) != 0. False otherwise.

Definition at line 2292 of file vstring.h.

3.1.2.6 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs) [inline]

Test difference of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

Definition at line 2305 of file vstring.h.

3.1.2.7 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*)

Concatenate two strings.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

Definition at line 180 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

3.1.2.8 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT ∗ *__lhs,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*)

Concatenate C string and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

Definition at line 193 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

3.1.2.9 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT *__lhs,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*)

Concatenate character and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

Definition at line 210 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

3.1.2.10 template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const _CharT * *__rhs*)

Concatenate string and C string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

Definition at line 223 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

3.1.2.11 template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, _CharT *__rhs*)

Concatenate string and character.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

Definition at line 240 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

3.1.2.12 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Test if string precedes string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2319 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.13 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const _CharT ∗ *__rhs*) `[inline]`

Test if string precedes C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2332 of file vstring.h.

3.1.2.14 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const _CharT ∗ *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Test if C string precedes string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2345 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.15 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Test if string doesn't follow string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't follow *__rhs.* False otherwise.

Definition at line 2399 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.16 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const _CharT * *__rhs*) `[inline]`

Test if string doesn't follow C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* doesn't follow *__rhs.* False otherwise.

Definition at line 2412 of file vstring.h.

3.1.2.17 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= (const _CharT * *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Test if C string doesn't follow string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* doesn't follow *__rhs.* False otherwise.

Definition at line 2425 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.18 template< typename _Tp > bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > & *__lhs*, const _Pointer_adapter< _Tp > & *__rhs*) `[inline]`

Comparison operators for _Pointer_adapter defer to the base class' comparison operators, when possible.

Definition at line 529 of file pointer.h.

3.1.2.19 template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Test equivalence of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

Definition at line 2228 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.20 template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator== (const _CharT * *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Test equivalence of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) == 0. False otherwise.

Definition at line 2252 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.21 template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const _CharT * *__rhs*) `[inline]`

Test equivalence of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

Definition at line 2265 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.22 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) [inline]

Test if string follows string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

Definition at line 2359 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.23 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, const _CharT ∗ *__rhs*) [inline]

Test if string follows C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

Definition at line 2372 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.24 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const _CharT ∗ *__lhs*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) [inline]

Test if C string follows string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* follows *__rhs.* False otherwise.

Definition at line 2385 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.25 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) [inline]

Test if string doesn't precede string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't precede *__rhs.* False otherwise.

Definition at line 2439 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.26 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs,* const _CharT ∗ *__rhs*) [inline]

Test if string doesn't precede C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* doesn't precede *__rhs.* False otherwise.

Definition at line 2452 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.27 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= (const _CharT ∗ *__lhs,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) [inline]

Test if C string doesn't precede string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

Definition at line 2465 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

3.1.2.28 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & *__lhs*, __versa_string< _CharT, _Traits, _Alloc, _Base > & *__rhs*) `[inline]`

Swap contents of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Exchanges the contents of *__lhs* and *__rhs* in constant time.

Definition at line 2479 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap().

3.2 __gnu_cxx::__detail Namespace Reference

Classes

- class __mini_vector

  __mini_vector<> is a stripped down version of the full-fledged std::vector<>.
- class _Bitmap_counter

  The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.
- class _Ffit_finder

  The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Enumerations

- enum { **_S_max_rope_depth** }
- enum { **bits_per_byte**, **bits_per_block** }
- enum **_Tag** { **_S_leaf**, **_S_concat**, **_S_substringfn**, **_S_function** }

Functions

- void __bit_allocate (size_t *__pbmap, size_t __pos) throw ()
- void __bit_free (size_t *__pbmap, size_t __pos) throw ()
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator **__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair >
  size_t __num_bitmaps (_AddrPair __ap)
- template<typename _AddrPair >
  size_t __num_blocks (_AddrPair __ap)

3.2.1 Detailed Description

Implementation details not part of the namespace __gnu_cxx interface.

3.2.2 Function Documentation

3.2.2.1 void __gnu_cxx::__detail::__bit_allocate (size_t ∗ *__pbmap*, size_t *__pos*) throw () `[inline]`

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 488 of file bitmap_allocator.h.

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object().

3.2.2.2 void __gnu_cxx::__detail::__bit_free (size_t ∗ *__pbmap*, size_t *__pos*) throw () `[inline]`

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 499 of file bitmap_allocator.h.

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object().

3.2.2.3 template<typename _AddrPair > size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair *__ap*) `[inline]`

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 276 of file bitmap_allocator.h.

References __num_blocks().

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object(), and __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object().

3.2.2.4 template<typename _AddrPair > size_t __gnu_cxx::__detail::__num_blocks (_AddrPair *__ap*) `[inline]`

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 268 of file bitmap_allocator.h.

Referenced by __num_bitmaps().

3.3 __gnu_cxx::typelist Namespace Reference

Functions

- template<typename Fn , typename Typelist >
  void **apply** (Fn &, Typelist)
- template<typename Gn , typename Typelist >
  void apply_generator (Gn &, Typelist)
- template<typename Gn , typename TypelistT , typename TypelistV >
  void **apply_generator** (Gn &, TypelistT, TypelistV)
- template<typename Fn , typename Typelist >
  void **apply_generator** (Fn &fn, Typelist)
- template<typename Fn , typename TypelistT , typename TypelistV >
  void **apply_generator** (Fn &fn, TypelistT, TypelistV)

3.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

3.3.2 Function Documentation

3.3.2.1 template<typename Gn , typename Typelist > void __gnu_cxx::typelist::apply_generator (Gn & , Typelist)

Apply all typelist types to generator functor.

3.4 __gnu_debug Namespace Reference

Classes

- class _After_nth_from
- struct _BeforeBeginHelper
- class _Equal_to
- class _Not_equal_to
- class _Safe_iterator

  Safe iterator wrapper.
- class _Safe_iterator_base

  Basic functionality for a safe iterator.
- class _Safe_local_iterator

  Safe iterator wrapper.
- class _Safe_local_iterator_base

  Basic functionality for a safe iterator.
- class _Safe_sequence

  Base class for constructing a safe sequence type that tracks iterators that reference it.
- class _Safe_sequence_base

  Base class that supports tracking of iterators that reference a sequence.
- class _Safe_unordered_container

  Base class for constructing a safe unordered container type that tracks iterators that reference it.
- class _Safe_unordered_container_base

  Base class that supports tracking of local iterators that reference an unordered container.
- class basic_string

  Class std::basic_string with safety/checking/debug instrumentation.

Typedefs

- typedef basic_string< char > **string**
- typedef basic_string< wchar_t > **wstring**

Enumerations

- enum **_Debug_msg_id** {
__msg_valid_range, **__msg_insert_singular**, **__msg_insert_different**, **__msg_erase_bad**,
__msg_erase_different, **__msg_subscript_oob**, **__msg_empty**, **__msg_unpartitioned**,
__msg_unpartitioned_pred, **__msg_unsorted**, **__msg_unsorted_pred**, **__msg_not_heap**,
__msg_not_heap_pred, **__msg_bad_bitset_write**, **__msg_bad_bitset_read**, **__msg_bad_bitset_flip**,
__msg_self_splice, **__msg_splice_alloc**, **__msg_splice_bad**, **__msg_splice_other**,
__msg_splice_overlap, **__msg_init_singular**, **__msg_init_copy_singular**, **__msg_init_const_singular**,
__msg_copy_singular, **__msg_bad_deref**, **__msg_bad_inc**, **__msg_bad_dec**,
__msg_iter_subscript_oob, **__msg_advance_oob**, **__msg_retreat_oob**, **__msg_iter_compare_bad**,
__msg_compare_different, **__msg_iter_order_bad**, **__msg_order_different**, **__msg_distance_bad**,
__msg_distance_different, **__msg_deref_istream**, **__msg_inc_istream**, **__msg_output_ostream**,
__msg_deref_istreambuf, **__msg_inc_istreambuf**, **__msg_insert_after_end**, **__msg_erase_after_bad**,
__msg_valid_range2, **__msg_local_iter_compare_bad**, **__msg_non_empty_range**, **__msg_self_move_assign**,
__msg_bucket_index_oob, **__msg_valid_load_factor**, **__msg_equal_allocs** }
- enum _Distance_precision { **__dp_equality**, **__dp_sign**, **__dp_exact** }

Functions

- template<typename _Iterator >
_Siter_base< _Iterator >
::iterator_type __base (_Iterator __it)
- template<typename _Iterator >
bool __check_dereferenceable (_Iterator &)
- template<typename _Tp >
bool __check_dereferenceable (const _Tp ∗__ptr)
- template<typename _Iterator , typename _Sequence >
bool __check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)
- template<typename _ForwardIterator , typename _Tp >
bool **__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
bool **__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator , typename _Tp >
bool **__check_partitioned_lower_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp >
bool **__check_partitioned_lower_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, std::random_access_iterator_tag __tag)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
bool **__check_partitioned_lower_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp , typename _Pred >
bool **__check_partitioned_lower_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, _Pred __pred, std::random_access_iterator_tag __tag)
- template<typename _ForwardIterator , typename _Tp >
bool **__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
bool **__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)

- template<typename _ForwardIterator , typename _Tp >
  bool **__check_partitioned_upper_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp >
  bool **__check_partitioned_upper_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, std::random_access_iterator_tag __tag)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
  bool **__check_partitioned_upper_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp , typename _Pred >
  bool **__check_partitioned_upper_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, _Pred __pred, std::random_access_iterator_tag __tag)
- template<typename _Iterator >
  bool **__check_singular** (_Iterator &)
- template<typename _Tp >
  bool __check_singular (const _Tp ∗__ptr)
- template<typename _Iterator , typename _Sequence >
  bool __check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)
- bool **__check_singular_aux** (const void ∗)
- bool __check_singular_aux (const _Safe_iterator_base ∗__x)
- template<typename _InputIterator >
  bool **__check_sorted** (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _InputIterator , typename _Predicate >
  bool **__check_sorted** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)
- template<typename _InputIterator >
  bool **__check_sorted_aux** (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _ForwardIterator >
  bool **__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence >
  bool **__check_sorted_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, std::random_access_iterator_tag __tag)
- template<typename _InputIterator , typename _Predicate >
  bool **__check_sorted_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)
- template<typename _ForwardIterator , typename _Predicate >
  bool **__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Predicate >
  bool **__check_sorted_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, _Predicate __pred, std::random_access_iterator_tag __tag)
- template<typename _InputIterator1 , typename _InputIterator2 >
  bool **__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _Predicate >
  bool **__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)
- template<typename _InputIterator >
  bool **__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)
- template<typename _InputIterator >
  bool **__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _InputIterator , typename _Predicate >
  bool **__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)

- template<typename _InputIterator , typename _Predicate >
  bool **__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)
- template<typename _CharT , typename _Integer >
  const _CharT ∗ __check_string (const _CharT ∗__s, const _Integer &__n __attribute__((__unused__)))
- template<typename _CharT >
  const _CharT ∗ __check_string (const _CharT ∗__s)
- template<typename _InputIterator >
  _InputIterator **__check_valid_range** (const _InputIterator &__first, const _InputIterator &__last __attribute__((__unused__)))
- template<typename _Iterator1 , typename _Iterator2 >
  std::pair< typename
  std::iterator_traits
  < _Iterator1 >
  ::difference_type,
  _Distance_precision > __get_distance (const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::random_access_iterator_tag)
- template<typename _Iterator1 , typename _Iterator2 >
  std::pair< typename
  std::iterator_traits
  < _Iterator1 >
  ::difference_type,
  _Distance_precision > **__get_distance** (const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::forward_iterator_tag)
- template<typename _Iterator1 , typename _Iterator2 >
  std::pair< typename
  std::iterator_traits
  < _Iterator1 >
  ::difference_type,
  _Distance_precision > **__get_distance** (const _Iterator1 &__lhs, const _Iterator2 &__rhs)
- template<typename _InputIterator >
  bool __valid_range (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _Iterator , typename _Sequence >
  bool __valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)
- template<typename _Iterator , typename _Sequence >
  bool __valid_range (const _Safe_local_iterator< _Iterator, _Sequence > &__first, const _Safe_local_iterator< _Iterator, _Sequence > &__last)
- template<typename _Integral >
  bool __valid_range_aux (const _Integral &, const _Integral &, std::__true_type)
- template<typename _InputIterator >
  bool __valid_range_aux (const _InputIterator &__first, const _InputIterator &__last, std::__false_type)
- template<typename _RandomAccessIterator >
  bool __valid_range_aux2 (const _RandomAccessIterator &__first, const _RandomAccessIterator &__last, std::random_access_iterator_tag)
- template<typename _InputIterator >
  bool __valid_range_aux2 (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_istream< _CharT,
  _Traits > & **getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_istream< _CharT,
  _Traits > & **getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)

- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator!=** (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator!=** (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator!=** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator!=** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator!=** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _Iterator , typename _Sequence >
  _Safe_iterator< _Iterator,
  _Sequence > **operator+** (typename _Safe_iterator< _Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _Sequence > &__i)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **operator+** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  _Safe_iterator< _IteratorL,
  _Sequence >::difference_type **operator-** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  _Safe_iterator< _Iterator,
  _Sequence >::difference_type **operator-** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator<** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator<** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)

- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator**< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator**< (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator**< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator<=** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator<=** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator<=** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator==** (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator==** (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator==** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator==** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator==** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator**> (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator**> (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator**> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)

- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator>** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator>=** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator>=** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator>=** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT , typename _Traits , typename _Allocator >
  void **swap** (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)

3.4.1 Detailed Description

GNU debug classes for public use.

3.4.2 Enumeration Type Documentation

3.4.2.1 enum __gnu_debug::_Distance_precision

The precision to which we can calculate the distance between two iterators.

Definition at line 70 of file safe_iterator.h.

3.4.3 Function Documentation

3.4.3.1 template<typename _Iterator > _Siter_base<_Iterator>::iterator_type __gnu_debug::__base (_Iterator *__it*) [inline]

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the __valid_range function thanks to the < operator.

Definition at line 546 of file functions.h.

Referenced by __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_before_dereferenceable(), std::boolalpha(), std::dec(), std::fixed(), std::hex(), std::internal(), std::left(), std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), std::nouppercase(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

3.4.3.2 template<typename _Iterator > bool __gnu_debug::__check_dereferenceable (_Iterator &) `[inline]`

Assume that some arbitrary iterator is dereferenceable, because we can't prove that it isn't.

Definition at line 70 of file functions.h.

3.4.3.3 template<typename _Tp > bool __gnu_debug::__check_dereferenceable (const _Tp * *__ptr*) `[inline]`

Non-NULL pointers are dereferenceable.

Definition at line 76 of file functions.h.

3.4.3.4 template<typename _Iterator , typename _Sequence > bool __gnu_debug::__check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > & *__x*) `[inline]`

Safe iterators know if they are singular.

Definition at line 82 of file functions.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable().

3.4.3.5 template<typename _Tp > bool __gnu_debug::__check_singular (const _Tp * *__ptr*) `[inline]`

Non-NULL pointers are nonsingular.

Definition at line 57 of file functions.h.

3.4.3.6 template<typename _Iterator , typename _Sequence > bool __gnu_debug::__check_singular (const _Safe_iterator< _Iterator, _Sequence > & *__x*) `[inline]`

Safe iterators know if they are singular.

Definition at line 63 of file functions.h.

References __gnu_debug::_Safe_iterator_base::_M_singular().

3.4.3.7 bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base * *__x*) `[inline]`

Iterators that derive from _Safe_iterator_base but that aren't _Safe_iterators can be determined singular or non-singular via _Safe_iterator_base.

Definition at line 64 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_singular().

3.4.3.8 template<typename _CharT , typename _Integer > const _CharT* __gnu_debug::__check_string (const _CharT * *__s*, const _Integer &__n *__attribute__(__unused__)*) `[inline]`

Checks that __s is non-NULL or __n == 0, and then returns __s.

Definition at line 168 of file functions.h.

3.4.3.9 template<typename _CharT > const _CharT* __gnu_debug::__check_string (const _CharT * *__s*) `[inline]`

Checks that __s is non-NULL and then returns __s.

Definition at line 180 of file functions.h.

3.4.3.10 template<typename _Iterator1 , typename _Iterator2 > std::pair<typename std::iterator_traits<_Iterator1>::difference_type, _Distance_precision> __gnu_debug::__get_distance (const _Iterator1 & *__lhs,* const _Iterator2 & *__rhs,* std::random_access_iterator_tag) [inline]

Determine the distance between two iterators with some known precision.

Definition at line 83 of file safe_iterator.h.

References std::make_pair().

3.4.3.11 template<typename _InputIterator > bool __gnu_debug::__valid_range (const _InputIterator & *__first,* const _InputIterator & *__last*) [inline]

Don't know what these iterators are, or if they are even iterators (we may get an integral type for InputIterator), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 131 of file functions.h.

References __valid_range_aux().

3.4.3.12 template<typename _Iterator , typename _Sequence > bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence > & *__first,* const _Safe_iterator< _Iterator, _Sequence > & *__last*) [inline]

Safe iterators know how to check if they form a valid range.

Definition at line 140 of file functions.h.

3.4.3.13 template<typename _Iterator , typename _Sequence > bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _Sequence > & *__first,* const _Safe_local_iterator< _Iterator, _Sequence > & *__last*) [inline]

Safe local iterators know how to check if they form a valid range.

Definition at line 147 of file functions.h.

3.4.3.14 template<typename _Integral > bool __gnu_debug::__valid_range_aux (const _Integral & *,* const _Integral & *,* std::__true_type) [inline]

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 111 of file functions.h.

Referenced by __valid_range().

3.4.3.15 template<typename _InputIterator > bool __gnu_debug::__valid_range_aux (const _InputIterator & *__first,* const _InputIterator & *__last,* std::__false_type) [inline]

We have iterators, so figure out what kind of iterators that are to see if we can check the range ahead of time.

Definition at line 119 of file functions.h.

References std::__iterator_category(), and __valid_range_aux2().

3.4.3.16 template<typename _RandomAccessIterator > bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator & *__first,* const _RandomAccessIterator & *__last,* std::random_access_iterator_tag) [inline]

If the distance between two random access iterators is nonnegative, assume the range is valid.

Definition at line 90 of file functions.h.

Referenced by __valid_range_aux().

3.4.3.17 template<typename _InputIterator > bool __gnu_debug::__valid_range_aux2 (const _InputIterator & , const _InputIterator & , std::input_iterator_tag) [inline]

Can't test for a valid range with input iterators, because iteration may be destructive. So we just assume that the range is valid.

Definition at line 101 of file functions.h.

3.5 __gnu_internal Namespace Reference

3.5.1 Detailed Description

GNU implemenation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

3.6 __gnu_parallel Namespace Reference

Classes

- struct __accumulate_binop_reduct

  General reduction, using a binary operator.
- struct __accumulate_selector

  std::accumulate() selector.
- struct __adjacent_difference_selector

  Selector that returns the difference between two adjacent __elements.
- struct __adjacent_find_selector

  Test predicate on two adjacent elements.
- class __binder1st

  Similar to std::binder1st, but giving the argument types explicitly.
- class __binder2nd

  Similar to std::binder2nd, but giving the argument types explicitly.
- struct __count_if_selector

  std::count_if () selector.
- struct __count_selector

  std::count() selector.
- struct __fill_selector

  std::fill() selector.
- struct __find_first_of_selector

  Test predicate on several elements.
- struct __find_if_selector

  Test predicate on a single element, used for std::find() and std::find_if ().
- struct __for_each_selector

  std::for_each() selector.
- struct __generate_selector

  std::generate() selector.
- struct __generic_find_selector

  Base class of all __gnu_parallel::__find_template selectors.
- struct __generic_for_each_selector

Generic __selector for embarrassingly parallel functions.
- struct __identity_selector

  Selector that just returns the passed iterator.
- struct __inner_product_selector

  std::inner_product() selector.
- struct __max_element_reduct

  Reduction for finding the maximum element, using a comparator.
- struct __min_element_reduct

  Reduction for finding the maximum element, using a comparator.
- struct __mismatch_selector

  Test inverted predicate on a single element.
- struct __multiway_merge_3_variant_sentinel_switch

  Switch for 3-way merging with __sentinels turned off.
- struct __multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for 3-way merging with __sentinels turned on.
- struct __multiway_merge_4_variant_sentinel_switch

  Switch for 4-way merging with __sentinels turned off.
- struct __multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for 4-way merging with __sentinels turned on.
- struct __multiway_merge_k_variant_sentinel_switch

  Switch for k-way merging with __sentinels turned on.
- struct __multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for k-way merging with __sentinels turned off.
- struct __replace_if_selector

  std::replace() selector.
- struct __replace_selector

  std::replace() selector.
- struct __transform1_selector

  std::transform() __selector, one input sequence variant.
- struct __transform2_selector

  std::transform() __selector, two input sequences variant.
- class __unary_negate

  Similar to std::unary_negate, but giving the argument types explicitly.
- struct _DRandomShufflingGlobalData

  Data known to every thread participating in __gnu_parallel::__parallel_random_shuffle().
- struct _DRSSorterPU

  Local data for a thread participating in __gnu_parallel::__parallel_random_shuffle().
- struct _DummyReduct

  Reduction function doing nothing.
- class _EqualFromLess

  Constructs predicate for equality from strict weak ordering predicate.
- struct _EqualTo

  Similar to std::equal_to, but allows two different types.
- class _GuardedIterator

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

- class _IteratorPair

  A pair of iterators. The usual iterator operations are applied to both child iterators.
- class _IteratorTriple

  A triple of iterators. The usual iterator operations are applied to all three child iterators.
- struct _Job

  One __job for a certain thread.
- struct _Less

  Similar to std::less, but allows two different types.
- class _Lexicographic

  Compare __a pair of types lexicographically, ascending.
- class _LexicographicReverse

  Compare __a pair of types lexicographically, descending.
- class _LoserTree

  Stable _LoserTree variant.
- class _LoserTree< false, _Tp, _Compare >

  Unstable _LoserTree variant.
- class _LoserTreeBase

  Guarded loser/tournament tree.
- class _LoserTreePointer

  Stable _LoserTree implementation.
- class _LoserTreePointer< false, _Tp, _Compare >

  Unstable _LoserTree implementation.
- class _LoserTreePointerBase

  Base class of _Loser Tree implementation using pointers.
- class _LoserTreePointerUnguarded

  Stable unguarded _LoserTree variant storing pointers.
- class _LoserTreePointerUnguarded< false, _Tp, _Compare >

  Unstable unguarded _LoserTree variant storing pointers.
- class _LoserTreePointerUnguardedBase

  Unguarded loser tree, keeping only pointers to the elements in the tree structure.
- struct _LoserTreeTraits

  Traits for determining whether the loser tree should use pointers or copies.
- class _LoserTreeUnguarded

  Stable implementation of unguarded _LoserTree.
- class _LoserTreeUnguarded< false, _Tp, _Compare >

  Non-Stable implementation of unguarded _LoserTree.
- class _LoserTreeUnguardedBase

  Base class for unguarded _LoserTree implementation.
- struct _Multiplies

  Similar to std::multiplies, but allows two different types.
- struct _Nothing

  Functor doing nothing.
- struct _Piece

  Subsequence description.
- struct _Plus

  Similar to std::plus, but allows two different types.

- struct _PMWMSSortingData

  Data accessed by all threads.
- class _PseudoSequence

  Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.
- class _PseudoSequenceIterator

  _Iterator associated with __gnu_parallel::_PseudoSequence. If features the usual random-access iterator functionality.
- struct _QSBThreadLocal

  Information local to one thread in the parallel quicksort run.
- class _RandomNumber

  Random number generator, based on the Mersenne twister.
- class _RestrictedBoundedConcurrentQueue

  Double-ended queue of bounded size, allowing lock-free atomic access. push_front() and pop_front() must not be called concurrently to each other, while pop_back() can be called concurrently at all times. `empty()`*,* `size()`*, and* `top()` *are intentionally not provided. Calling them would not make sense in a concurrent setting.*
- struct _SamplingSorter

  Stable sorting functor.
- struct _SamplingSorter< false, _RAIter, _StrictWeakOrdering >

  Non-__stable sorting functor.
- struct _Settings

  class _Settings Run-time settings for the parallel mode including all tunable parameters.
- struct _SplitConsistently

  Split consistently.
- struct _SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >

  Split by sampling.
- struct _SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

  Split by exact splitting.
- struct balanced_quicksort_tag

  Forces parallel sorting using balanced quicksort at compile time.
- struct balanced_tag

  Recommends parallel execution using dynamic load-balancing at compile time.
- struct constant_size_blocks_tag

  Selects the constant block size variant for std::find().
- struct default_parallel_tag

  Recommends parallel execution using the default parallel algorithm.
- struct equal_split_tag

  Selects the equal splitting variant for std::find().
- struct exact_tag

  Forces parallel merging with exact splitting, at compile time.
- struct find_tag

  Base class for for std::find() variants.
- struct growing_blocks_tag

  Selects the growing block size variant for std::find().
- struct multiway_mergesort_exact_tag

  Forces parallel sorting using multiway mergesort with exact splitting at compile time.
- struct multiway_mergesort_sampling_tag

  Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.
- struct multiway_mergesort_tag

Forces parallel sorting using multiway mergesort at compile time.

- struct omp_loop_static_tag

  Recommends parallel execution using OpenMP static load-balancing at compile time.
- struct omp_loop_tag

  Recommends parallel execution using OpenMP dynamic load-balancing at compile time.
- struct parallel_tag

  Recommends parallel execution at compile time, optionally using a user-specified number of threads.
- struct quicksort_tag

  Forces parallel sorting using unbalanced quicksort at compile time.
- struct sampling_tag

  Forces parallel merging with exact splitting, at compile time.
- struct sequential_tag

  Forces sequential execution at compile time.
- struct unbalanced_tag

  Recommends parallel execution using static load-balancing at compile time.

Typedefs

- typedef unsigned short _BinIndex
- typedef int64_t _CASable
- typedef uint64_t _SequenceIndex
- typedef uint16_t _ThreadIndex

Enumerations

- enum _AlgorithmStrategy { **heuristic**, **force_sequential**, **force_parallel** }
- enum _FindAlgorithm { **GROWING_BLOCKS**, **CONSTANT_SIZE_BLOCKS**, **EQUAL_SPLIT** }
- enum _MultiwayMergeAlgorithm { **LOSER_TREE** }
- enum _Parallelism {
  sequential, parallel_unbalanced, parallel_balanced, parallel_omp_loop,
  parallel_omp_loop_static, parallel_taskqueue }
- enum _PartialSumAlgorithm { **RECURSIVE**, **LINEAR** }
- enum _SortAlgorithm { **MWMS**, **QS**, **QS_BALANCED** }
- enum _SplittingAlgorithm { **SAMPLING**, **EXACT** }

Functions

- template<typename _Tp >
  _Tp **__add_omp** (volatile _Tp ∗__ptr, _Tp __addend)
- template<typename _RAIter , typename _DifferenceTp >
  void __calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp ∗__off)
- template<typename _Tp >
  bool **__cas_omp** (volatile _Tp ∗__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp >
  bool __compare_and_swap (volatile _Tp ∗__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **__copy_tail** (std::pair< _IIter, _IIter > __b, std::pair< _IIter, _IIter > __e, _OutputIterator __r)
- void __decode2 (_CASable __x, int &__a, int &__b)

- template<typename _RAIter , typename _DifferenceTp >
  void __determine_samples (_PMWMSSortingData< _RAIter > ∗__sd, _DifferenceTp __num_samples)
- _CASable __encode2 (int __a, int __b)
- template<typename _DifferenceType , typename _OutputIterator >
  _OutputIterator __equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)
- template<typename _DifferenceType >
  _DifferenceType __equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)
- template<typename _Tp >
  _Tp __fetch_and_add (volatile _Tp ∗__ptr, _Tp __addend)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)
- template<typename _IIter , typename _UserOp , typename _Functionality , typename _Red , typename _Result >
  _UserOp __for_each_template_random_access (_IIter __begin, _IIter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator_traits< _IIter >::difference_type __bound, _Parallelism __parallelism_tag)
- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- _ThreadIndex **__get_max_threads** ()
- bool **__is_parallel** (const _Parallelism __p)
- template<typename _IIter , typename _Compare >
  bool __is_sorted (_IIter __begin, _IIter __end, _Compare __comp)
- template<typename _RAIter , typename _Compare >
  _RAIter __median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)
- template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)

- template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare >
  _RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template<typename _RAIter1 , typename _RAIter3 , typename _Compare >
  _RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template<typename _RAIter , typename _Compare >
  void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter , typename _Compare >
  void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator __parallel_partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator __parallel_partial_sum_basecase (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::value_type __value)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator __parallel_partial_sum_linear (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::difference_type __n)
- template<typename _RAIter , typename _Predicate >
  std::iterator_traits< _RAIter >
  ::difference_type __parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__parallel_set_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__parallel_set_intersection** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _Operation >
  _OutputIterator **__parallel_set_operation** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Operation __op)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__parallel_set_symmetric_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__parallel_set_union** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<bool __stable, typename _RAIter , typename _Compare , typename _Parallelism >
  void **__parallel_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)

- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)
- template<typename _RAIter , typename _Compare >
  void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  std::iterator_traits< _RAIter >
  ::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _IIter , class _OutputIterator , class _BinaryPredicate >
  _OutputIterator __parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _IIter , class _OutputIterator >
  _OutputIterator __parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result)
- template<typename _RAIter , typename _Compare >
  void __qsb_conquer (_QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter , typename _Compare >
  std::iterator_traits< _RAIter >
  ::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)
- template<typename _RandomNumberGenerator >
  int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Size >
  _Size __rd_log2 (_Size __n)
- template<typename _Tp >
  _Tp __round_up_to_pow2 (_Tp __x)
- template<typename __RAIter1 , typename __RAIter2 , typename _Pred >
  __RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)

- template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)
- template<typename _IIter >
  void __shrink (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length)
- template<typename _IIter >
  void __shrink_and_double (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)
- void __yield ()
- template<typename _IIter , typename _FunctorType >
  size_t list_partition (const _IIter __begin, const _IIter __end, _IIter ∗__starts, size_t ∗__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)
- template<typename _Tp >
  const _Tp & max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp >
  const _Tp & min (const _Tp &__a, const _Tp &__b)
- template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare >
  void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())
- template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare >
  _Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
  void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > ∗__pieces)

- template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
  void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Splitter , typename _Compare >
  _RAIter3 parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
  void parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
  void parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)

- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)

Variables

- static const int _CASable_bits
- static const _CASable _CASable_mask

3.6.1 Detailed Description

GNU parallel code for public use.

3.6.2 Typedef Documentation

3.6.2.1 typedef unsigned short __gnu_parallel::_BinIndex

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file random_shuffle.h.

3.6.2.2 typedef int64_t __gnu_parallel::_CASable

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file types.h.

3.6.2.3 typedef uint64_t __gnu_parallel::_SequenceIndex

Unsigned integer to index __elements. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file types.h.

3.6.2.4 typedef uint16_t __gnu_parallel::_ThreadIndex

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file types.h.

3.6.3 Enumeration Type Documentation

3.6.3.1 enum __gnu_parallel::_AlgorithmStrategy

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

3.6.3.2 enum __gnu_parallel::_FindAlgorithm

Find algorithms:

Definition at line 106 of file types.h.

3.6.3.3 enum __gnu_parallel::_MultiwayMergeAlgorithm

Merging algorithms:

Definition at line 85 of file types.h.

3.6.3.4 enum __gnu_parallel::_Parallelism

Run-time equivalents for the compile-time tags.

Enumerator:

sequential Not parallel.
parallel_unbalanced Parallel unbalanced (equal-sized chunks).
parallel_balanced Parallel balanced (work-stealing).
parallel_omp_loop Parallel with OpenMP dynamic load-balancing.
parallel_omp_loop_static Parallel with OpenMP static load-balancing.
parallel_taskqueue Parallel with OpenMP taskqueue construct.

Definition at line 44 of file types.h.

3.6.3.5 enum __gnu_parallel::_PartialSumAlgorithm

Partial sum algorithms: recursive, linear.

Definition at line 91 of file types.h.

3.6.3.6 enum __gnu_parallel::_SortAlgorithm

Sorting algorithms:

Definition at line 76 of file types.h.

3.6.3.7 enum __gnu_parallel::_SplittingAlgorithm

Sorting/merging algorithms: sampling, __exact.

Definition at line 98 of file types.h.

3.6.4 Function Documentation

3.6.4.1 template<typename _RAIter , typename _DifferenceTp > void __gnu_parallel::__calc_borders (_RAIter *__elements,* _DifferenceTp *__length,* _DifferenceTp * *__off*)

Precalculate __advances for Knuth-Morris-Pratt algorithm.

Parameters

__elements	Begin iterator of sequence to search for.
__length	Length of sequence to search for.
__off	Returned __offsets.

Definition at line 51 of file search.h.

Referenced by __search_template().

3.6.4.2 template<typename _Tp > bool __gnu_parallel::__compare_and_swap (volatile _Tp * *__ptr,* _Tp *__comparand,* _Tp *__replacement*) [inline]

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Parameters

__ptr	Pointer to signed integer.
__comparand	Compare value.
__replacement	Replacement value.

Definition at line 108 of file parallel/compatibility.h.

Referenced by __parallel_partition(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front().

3.6.4.3 void __gnu_parallel::__decode2 (_CASable *__x,* int & *__a,* int & *__b*) [inline]

Decode two integers from one gnu_parallel::_CASable.

Parameters

__x	__gnu_parallel::_CASable to decode integers from.
__a	First integer, to be decoded from the most-significant `_CASable_bits/2` bits of `__x`.
__b	Second integer, to be encoded in the least-significant `_CASable_bits/2` bits of `__x`.

See Also

__encode2

Definition at line 133 of file parallel/base.h.

References _CASable_bits, and _CASable_mask.

Referenced by __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front().

3.6.4.4 template<typename _RAIter , typename _DifferenceTp > void __gnu_parallel::__determine_samples (_PMWMSSortingData< _RAIter > * __sd, _DifferenceTp __num_samples)

Select _M_samples from a sequence.

Parameters

__sd	Pointer to algorithm data. _Result will be placed in `__sd->_M_samples`.
__num_samples	Number of _M_samples to select.

Definition at line 97 of file multiway_mergesort.h.

References __equally_split(), __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, and __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts.

3.6.4.5 _CASable __gnu_parallel::__encode2 (int __a, int __b) `[inline]`

Encode two integers into one gnu_parallel::_CASable.

Parameters

__a	First integer, to be encoded in the most-significant `_CASable_bits/2` bits.
__b	Second integer, to be encoded in the least-significant `_CASable_bits/2` bits.

Returns

value encoding `__a` and `__b`.

See Also

__decode2

Definition at line 119 of file parallel/base.h.

References _CASable_bits.

Referenced by __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::_RestrictedBoundedConcurrentQueue(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_back(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::pop_front(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front().

3.6.4.6 template<typename _DifferenceType , typename _OutputIterator > _OutputIterator __gnu_parallel::__equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)

function to split a sequence into parts of almost equal size.

The resulting sequence __s of length __num_threads+1 contains the splitting positions when splitting the range [0,__n) into parts of almost equal size (plus minus 1). The first entry is 0, the last one n. There may result empty parts.

Parameters

__n	Number of elements
__num_threads	Number of parts
__s	Splitters

Returns

End of __splitter sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file equally_split.h.

Referenced by __determine_samples(), __find_template(), __parallel_partial_sum_linear(), __parallel_unique_copy(), __search_template(), and multiway_merge_exact_splitting().

3.6.4.7 template<typename _DifferenceType > _DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType *__n*, _ThreadIndex *__num_threads*, _ThreadIndex *__thread_no*)

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number __thread_no (included) and thread number __thread_no+1 (excluded).

Parameters

__n	Number of elements
__num_threads	Number of parts
__thread_no	Number of threads

Returns

splitting point

Definition at line 75 of file equally_split.h.

Referenced by __for_each_template_random_access_ed().

3.6.4.8 template<typename _Tp > _Tp __gnu_parallel::__fetch_and_add (volatile _Tp * *__ptr*, _Tp *__addend*) `[inline]`

Add a value to a variable, atomically.

Parameters

__ptr	Pointer to a signed integer.
__addend	Value to add.

Definition at line 74 of file parallel/compatibility.h.

Referenced by __parallel_partition(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Piece >::push_front().

3.6.4.9 template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (_RAIter1 *__begin1*, _RAIter1 *__end1*, _RAIter2 *__begin2*, _Pred *__pred*, _Selector *__selector*) `[inline]`

Parallel std::find, switch for different algorithms.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

Definition at line 60 of file find.h.

References __gnu_parallel::_Settings::get(), and std::make_pair().

3.6.4.10 template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred,* _Selector *__selector,* equal_split_tag)

Parallel std::find, equal splitting variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

Definition at line 97 of file find.h.

References __equally_split(), and _GLIBCXX_CALL.

3.6.4.11 template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred,* _Selector *__selector,* growing_blocks_tag)

Parallel std::find, growing block size variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See Also

__gnu_parallel::_Settings::find_sequential_search_size
__gnu_parallel::_Settings::find_scale_factor

There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file find.h.

References _GLIBCXX_CALL, __gnu_parallel::_Settings::find_scale_factor, __gnu_parallel::_Settings::find_sequential_search_size, and __gnu_parallel::_Settings::get().

3.6.4.12 template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred,* _Selector *__selector,* constant_size_blocks_tag)

Parallel std::find, constant block size variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See Also

__gnu_parallel::_Settings::find_sequential_search_size
__gnu_parallel::_Settings::find_block_size There are two main differences between the growing blocks and the constant-size blocks variants. 1. For GB, the block size grows; for CSB, the block size is fixed. 2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file find.h.

References _GLIBCXX_CALL, __gnu_parallel::_Settings::find_initial_block_size, __gnu_parallel::_Settings::find_sequential_search_size, and __gnu_parallel::_Settings::get().

3.6.4.13 template<typename _IIter , typename _UserOp , typename _Functionality , typename _Red , typename _Result > _UserOp __gnu_parallel::__for_each_template_random_access (_IIter *__begin,* _IIter *__end,* _UserOp *__user_op,* _Functionality & *__functionality,* _Red *__reduction,* _Result *__reduction_start,* _Result & *__output,* typename std::iterator_traits< _IIter >::difference_type *__bound,* _Parallelism *__parallelism_tag*)

Chose the desired algorithm by evaluating `__parallelism_tag`.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__user_op	A user-specified functor (comparator, predicate, associative operator,...)
__functionality	functor to *process* an element with __user_op (depends on desired functionality, e. g. accumulate, for_each,...
__reduction	Reduction functor.
__reduction_start	Initial value for reduction.
__output	Output iterator.
__bound	Maximum number of elements processed.
__parallelism_tag	Parallelization method

Definition at line 61 of file for_each.h.

References __for_each_template_random_access_ed(), __for_each_template_random_access_omp_loop(), __for_each_template_random_access_workstealing(), parallel_omp_loop, parallel_omp_loop_static, and parallel_unbalanced.

3.6.4.14 template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter *__begin,* _RAIter *__end,* _Op *__o,* _Fu & *__f,* _Red *__r,* _Result *__base,* _Result & *__output,* typename std::iterator_traits< _RAIter >::difference_type *__bound*)

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, ...)
__f	Functor to "process" an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to "add" a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file par_loop.h.

References __equally_split_point().

Referenced by __for_each_template_random_access().

3.6.4.15 template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter *__begin,* _RAIter *__end,* _Op *__o,* _Fu & *__f,* _Red *__r,* _Result *__base,* _Result & *__output,* typename std::iterator_traits< _RAIter >::difference_type *__bound*)

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, etc.).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file omp_loop.h.

Referenced by __for_each_template_random_access().

3.6.4.16 template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter *__begin,* _RAIter *__end,* _Op *__o,* _Fu & *__f,* _Red *__r,* _Result *__base,* _Result & *__output,* typename std::iterator_traits< _RAIter >::difference_type *__bound*)

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, ...).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed __elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file omp_loop_static.h.

3.6.4.17 template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter *__begin,* _RAIter *__end,* _Op *__op,* _Fu & *__f,* _Red *__r,* _Result *__base,* _Result & *__output,* typename std::iterator_traits< _RAIter >::difference_type *__bound*)

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__op	User-supplied functor (comparator, predicate, adding functor, ...).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file workstealing.h.

References __gnu_debug::__base(), __yield(), _GLIBCXX_CALL, __gnu_parallel::_Job< _DifferenceTp >::_M_first, __gnu_parallel::_Job< _DifferenceTp >::_M_last, __gnu_parallel::_Job< _DifferenceTp >::_M_load, __gnu_parallel::_Settings::cache_line_size, __gnu_parallel::_Settings::get(), and min().

Referenced by __for_each_template_random_access().

3.6.4.18 template<typename _IIter , typename _Compare > bool __gnu_parallel::__is_sorted (_IIter *__begin,* _IIter *__end,* _Compare *__comp*)

Check whether [__begin, `__end`) is sorted according to `__comp`.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.

Returns

`true` if sorted, `false` otherwise.

Definition at line 51 of file checkers.h.

Referenced by __sequential_multiway_merge(), multiway_merge_loser_tree_sentinel(), and parallel_multiway_merge().

3.6.4.19 template<typename _RAIter , typename _Compare > _RAIter __gnu_parallel::__median_of_three_iterators (_RAIter *__a,* _RAIter *__b,* _RAIter *__c,* _Compare *__comp*)

Compute the median of three referenced elements, according to `__comp`.

Parameters

__a	First iterator.
__b	Second iterator.
__c	Third iterator.
__comp	Comparator.

Definition at line 398 of file parallel/base.h.

Referenced by __qsb_divide().

3.6.4.20 template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare > _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 & *__begin1,* _RAIter1 *__end1,* _RAIter2 & *__begin2,* _RAIter2 *__end2,* _OutputIterator *__target,* _DifferenceTp *__max_length,* _Compare *__comp*) `[inline]`

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Definition at line 171 of file merge.h.

References __merge_advance_movc(), and _GLIBCXX_CALL.

Referenced by __parallel_merge_advance(), and __sequential_multiway_merge().

3.6.4.21 template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare > _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 & *__begin1,* _RAIter1 *__end1,* _RAIter2 & *__begin2,* _RAIter2 *__end2,* _OutputIterator *__target,* _DifferenceTp *__max_length,* _Compare *__comp*)

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Definition at line 105 of file merge.h.

Referenced by __merge_advance().

3.6.4.22 template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare > _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 & *__begin1,* _RAIter1 *__end1,* _RAIter2 & *__begin2,* _RAIter2 *__end2,* _OutputIterator *__target,* _DifferenceTp *__max_length,* _Compare *__comp*)

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Definition at line 57 of file merge.h.

3.6.4.23 template< typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 & *__begin1,* _RAIter1 *__end1,* _RAIter2 & *__begin2,* _RAIter2 *__end2,* _RAIter3 *__target,* typename std::iterator_traits< _RAIter1 >::difference_type *__max_length,* _Compare *__comp*) `[inline]`

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Definition at line 195 of file merge.h.

References __merge_advance().

3.6.4.24 template< typename _RAIter1 , typename _RAIter3 , typename _Compare > _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 & *__begin1,* _RAIter1 *__end1,* _RAIter1 & *__begin2,* _RAIter1 *__end2,* _RAIter3 *__target,* typename std::iterator_traits< _RAIter1 >::difference_type *__max_length,* _Compare *__comp*) `[inline]`

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto parallel_multiway_merge.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.

__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Definition at line 223 of file merge.h.

References std::make_pair(), multiway_merge_exact_splitting(), and parallel_multiway_merge().

3.6.4.25 template<typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_nth_element (_RAIter *__begin,* _RAIter *__nth,* _RAIter *__end,* _Compare *__comp*)

Parallel implementation of std::nth_element().

Parameters

__begin	Begin iterator of input sequence.
__nth	_Iterator of element that must be in position afterwards.
__end	End iterator of input sequence.
__comp	Comparator.

Definition at line 332 of file partition.h.

References __parallel_partition(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), std::max(), __gnu_parallel::_Settings::nth_element_minimal_n, and __gnu_parallel::_Settings::partition_minimal_n.

Referenced by __parallel_partial_sort().

3.6.4.26 template<typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_partial_sort (_RAIter *__begin,* _RAIter *__middle,* _RAIter *__end,* _Compare *__comp*)

Parallel implementation of std::partial_sort().

Parameters

__begin	Begin iterator of input sequence.
__middle	Sort until this position.
__end	End iterator of input sequence.
__comp	Comparator.

Definition at line 422 of file partition.h.

References __parallel_nth_element().

3.6.4.27 template<typename _IIter , typename _OutputIterator , typename _BinaryOperation > _OutputIterator __gnu_parallel::__parallel_partial_sum (_IIter *__begin,* _IIter *__end,* _OutputIterator *__result,* _BinaryOperation *__bin_op*)

Parallel partial sum front-__end.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.

__result	Begin iterator of output sequence.
__bin_op	Associative binary function.

Returns

End iterator of output sequence.

Definition at line 205 of file partial_sum.h.

References __parallel_partial_sum_linear(), _GLIBCXX_CALL, and __gnu_parallel::_Settings::get().

3.6.4.28 template<typename _IIter , typename _OutputIterator , typename _BinaryOperation > _OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_IIter *__begin,* _IIter *__end,* _OutputIterator *__result,* _BinaryOperation *__bin_op,* typename std::iterator_traits< _IIter >::value_type *__value*)

Base case prefix sum routine.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.
__value	Start value. Must be passed since the neutral element is unknown in general.

Returns

End iterator of output sequence.

Definition at line 58 of file partial_sum.h.

Referenced by __parallel_partial_sum_linear().

3.6.4.29 template<typename _IIter , typename _OutputIterator , typename _BinaryOperation > _OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_IIter *__begin,* _IIter *__end,* _OutputIterator *__result,* _BinaryOperation *__bin_op,* typename std::iterator_traits< _IIter >::difference_type *__n*)

Parallel partial sum implementation, two-phase approach, no recursion.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.
__n	Length of sequence.

Returns

End iterator of output sequence.

Definition at line 89 of file partial_sum.h.

References __equally_split(), __parallel_partial_sum_basecase(), std::accumulate(), __gnu_parallel::_Settings::get(), and __gnu_parallel::_Settings::partial_sum_dilation.

Referenced by __parallel_partial_sum().

3.6.4.30 template<typename _RAIter , typename _Predicate > std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_partition (_RAIter *__begin,* _RAIter *__end,* _Predicate *__pred,* _ThreadIndex *__num_threads*)

Parallel implementation of std::partition.

Parameters

__begin	Begin iterator of input sequence to split.
__end	End iterator of input sequence to split.
__pred	Partition predicate, possibly including some kind of pivot.
__num_threads	Maximum number of threads to use for this task.

Returns

Number of elements not fulfilling the predicate.

Definition at line 56 of file partition.h.

References __compare_and_swap(), __fetch_and_add(), _GLIBCXX_CALL, _GLIBCXX_VOLATILE, __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::partition_chunk_share, and __gnu_parallel::_Settings::partition_chunk_size.

Referenced by __parallel_nth_element(), __parallel_sort_qs_divide(), and __qsb_divide().

3.6.4.31 template<typename _RAIter , typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle (_RAIter *__begin,* _RAIter *__end,* _RandomNumberGenerator *__rng* = `_RandomNumber()`) `[inline]`

Parallel random public call.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__rng	Random number generator to use.

Definition at line 522 of file random_shuffle.h.

References __parallel_random_shuffle_drs().

3.6.4.32 template<typename _RAIter , typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter *__begin,* _RAIter *__end,* typename std::iterator_traits< _RAIter >::difference_type *__n,* _ThreadIndex *__num_threads,* _RandomNumberGenerator & *__rng*)

Main parallel random shuffle step.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__n	Length of sequence.
__num_threads	Number of threads to use.
__rng	Random number generator to use.

Definition at line 265 of file random_shuffle.h.

References __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end, __parallel_random_shuffle_drs_pu(), __rd_log2(), __round_up_to_pow2(), __sequential_random_shuffle(), _GLIBCXX_CALL, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bin_proc, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_sd, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries, __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::L2_cache_size, std::min(), and __gnu_parallel::_Settings::TLB_size.

Referenced by __parallel_random_shuffle().

3.6.4.33 template<typename _RAIter , typename _RandomNumberGenerator > void __gnu_parallel::__parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > ∗ *__pus*)

Random shuffle code executed by each thread.

Parameters

__pus	Array of thread-local data records.

Definition at line 122 of file random_shuffle.h.

References __random_number_pow2(), __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_sd, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts, and std::partial_sum().

Referenced by __parallel_random_shuffle_drs().

3.6.4.34 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin,* _RAIter *__end,* _Compare *__comp,* multiway_mergesort_tag *__parallelism*) [inline]

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

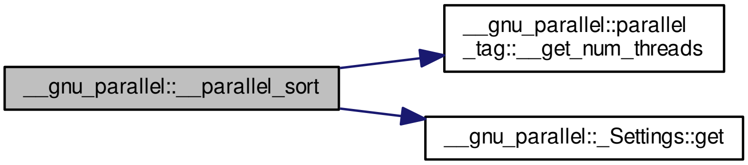
Template Parameters

__stable	Sort stable.

Definition at line 75 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, and __gnu_parallel::_Settings::get().

Here is the call graph for this function:



3.6.4.35 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, multiway_mergesort_exact_tag *__parallelism*) [inline]

Choose multiway mergesort with exact splitting, for parallel sorting.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

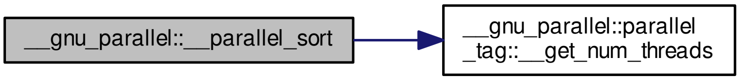
Template Parameters

__stable	Sort stable.

Definition at line 99 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), and _GLIBCXX_CALL.

Here is the call graph for this function:



3.6.4.36 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, multiway_mergesort_sampling_tag *__parallelism*) [inline]

Choose multiway mergesort with splitting by sampling, for parallel sorting.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

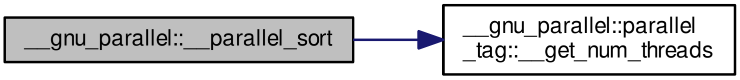
Template Parameters

__stable	Sort stable.

Definition at line 120 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), and _GLIBCXX_CALL.

Here is the call graph for this function:



3.6.4.37 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, quicksort_tag *__parallelism*) [inline]

Choose quicksort for parallel sorting.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

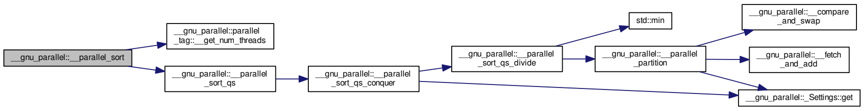
Template Parameters

__stable	Sort stable.

Definition at line 140 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qs(), and _GLIBCXX_CALL.

Here is the call graph for this function:



3.6.4.38 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin,* _RAIter *__end,* _Compare *__comp,* balanced_quicksort_tag *__parallelism*) [inline]

Choose balanced quicksort for parallel sorting.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

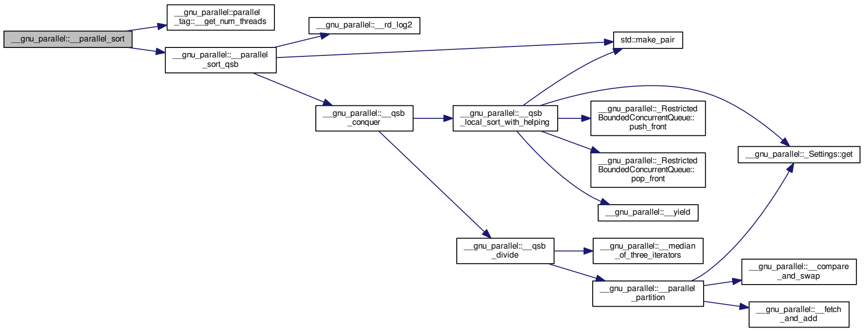
Template Parameters

__stable	Sort stable.

Definition at line 161 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qsb(), and _GLIBCXX_CALL.

Here is the call graph for this function:



3.6.4.39 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin,* _RAIter *__end,* _Compare *__comp,* default_parallel_tag *__parallelism*) [inline]

Choose multiway mergesort with exact splitting, for parallel sorting.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

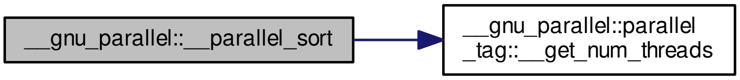
Template Parameters

__stable	Sort stable.

Definition at line 183 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), and _GLIBCXX_CALL.

Here is the call graph for this function:



3.6.4.40 template<bool __stable, typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, parallel_tag *__parallelism*) [inline]

Choose a parallel sorting algorithm.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

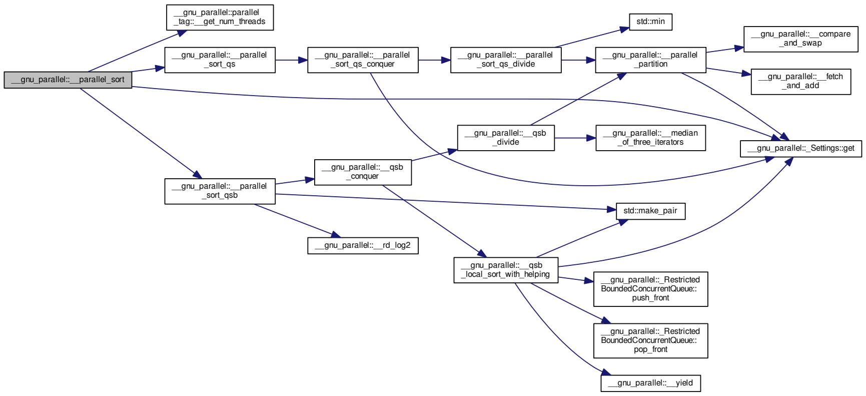
Template Parameters

__stable	Sort stable.

Definition at line 203 of file sort.h.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qs(), __parallel_sort_qsb(), _GLIBCXX_CALL, and __gnu_parallel::_Settings::get().

Here is the call graph for this function:



3.6.4.41 template<typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort_qs (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, _ThreadIndex *__num_threads*)

Unbalanced quicksort main call.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator input sequence, ignored.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

Definition at line 154 of file quicksort.h.

References __parallel_sort_qs_conquer(), and _GLIBCXX_CALL.

Referenced by __parallel_sort().

3.6.4.42 template<typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, _ThreadIndex *__num_threads*)

Unbalanced quicksort conquer step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

Definition at line 101 of file quicksort.h.

References __parallel_sort_qs_divide(), and __gnu_parallel::_Settings::get().

Referenced by __parallel_sort_qs().

3.6.4.43 template<typename _RAIter , typename _Compare > std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, typename std::iterator_traits< _RAIter >::difference_type *__pivot_rank*, typename std::iterator_traits< _RAIter >::difference_type *__num_samples*, _ThreadIndex *__num_threads*)

Unbalanced quicksort divide step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__pivot_rank	Desired __rank of the pivot.
__num_samples	Choose pivot from that many samples.
__num_threads	Number of threads that are allowed to work on this part.

Definition at line 51 of file quicksort.h.

References __parallel_partition(), and std::min().

Referenced by __parallel_sort_qs_conquer().

3.6.4.44 template<typename _RAIter , typename _Compare > void __gnu_parallel::__parallel_sort_qsb (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, _ThreadIndex *__num_threads*)

Top-level quicksort routine.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

Definition at line 430 of file balanced_quicksort.h.

References __qsb_conquer(), __rd_log2(), _GLIBCXX_CALL, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_-elements_leftover, and std::make_pair().

Referenced by __parallel_sort().

3.6.4.45 template<typename _IIter , class _OutputIterator , class _BinaryPredicate > _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter *__first*, _IIter *__last*, _OutputIterator *__result*, _BinaryPredicate *__binary_pred*)

Parallel std::unique_copy(), w/__o explicit equality predicate.

Parameters

__first	Begin iterator of input sequence.
__last	End iterator of input sequence.
__result	Begin iterator of result __sequence.
__binary_pred	Equality predicate.

Returns

End iterator of result __sequence.

Definition at line 50 of file unique_copy.h.

References __equally_split(), and _GLIBCXX_CALL.

Referenced by __parallel_unique_copy().

3.6.4.46 template<typename _IIter , class _OutputIterator > _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter *__first*, _IIter *__last*, _OutputIterator *__result*) `[inline]`

Parallel std::unique_copy(), without explicit equality predicate.

Parameters

__first	Begin iterator of input sequence.
__last	End iterator of input sequence.
__result	Begin iterator of result __sequence.

Returns

End iterator of result __sequence.

Definition at line 186 of file unique_copy.h.

References __parallel_unique_copy().

3.6.4.47 template<typename _RAIter , typename _Compare > void __gnu_parallel::__qsb_conquer (_QSBThreadLocal< _RAIter > ** *__tls,* _RAIter *__begin,* _RAIter *__end,* _Compare *__comp,* _ThreadIndex *__iam,* _ThreadIndex *__num_threads,* bool *__parent_wait*)

Quicksort conquer step.

Parameters

__tls	Array of thread-local storages.
__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__iam	Number of the thread processing this function.
__num_threads	Number of threads that are allowed to work on this part.

Definition at line 171 of file balanced_quicksort.h.

References __qsb_divide(), __qsb_local_sort_with_helping(), __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover, and __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial.

Referenced by __parallel_sort_qsb().

3.6.4.48 template<typename _RAIter , typename _Compare > std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__qsb_divide (_RAIter *__begin,* _RAIter *__end,* _Compare *__comp,* _ThreadIndex *__num_threads*)

Balanced quicksort divide step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

Precondition

(__end-__begin)>=1

Definition at line 100 of file balanced_quicksort.h.

References __median_of_three_iterators(), and __parallel_partition().

Referenced by __qsb_conquer().

3.6.4.49 template<typename _RAIter , typename _Compare > void __gnu_parallel::__qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > ** *__tls,* _Compare & *__comp,* _ThreadIndex *__iam,* bool *__wait*)

Quicksort step doing load-balanced local sort.

Parameters

__tls	Array of thread-local storages.
__comp	Comparator.
__iam	Number of the thread processing this function.

Definition at line 247 of file balanced_quicksort.h.

References __yield(), _GLIBCXX_ASSERTIONS, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_leftover_parts, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads, __gnu_parallel::_Settings::get(), std::make_pair(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front(), and __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n.

Referenced by __qsb_conquer().

3.6.4.50 template<typename _RandomNumberGenerator > int __gnu_parallel::__random_number_pow2 (int *__logp*, _RandomNumberGenerator & *__rng*) `[inline]`

Generate a random number in [0,2^__logp).

Parameters

__logp	Logarithm (basis 2) of the upper range __bound.
__rng	Random number generator to use.

Definition at line 115 of file random_shuffle.h.

Referenced by __parallel_random_shuffle_drs_pu(), and __sequential_random_shuffle().

3.6.4.51 template<typename _Size > _Size __gnu_parallel::__rd_log2 (_Size *__n*) `[inline]`

Calculates the rounded-down logarithm of `__n` for base 2.

Parameters

__n	Argument.

Returns

Returns 0 for any argument <1.

Definition at line 102 of file parallel/base.h.

Referenced by __parallel_random_shuffle_drs(), __parallel_sort_qsb(), __round_up_to_pow2(), __sequential_random_shuffle(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase(), multiseq_partition(), and multiseq_selection().

3.6.4.52 template<typename _Tp > _Tp __gnu_parallel::__round_up_to_pow2 (_Tp *__x*)

Round up to the next greater power of 2.

Parameters

__x	_Integer to round up

Definition at line 248 of file random_shuffle.h.

References __rd_log2().

Referenced by __parallel_random_shuffle_drs(), __sequential_random_shuffle(), and multiseq_selection().

3.6.4.53 template<typename __RAIter1 , typename __RAIter2 , typename _Pred > __RAIter1 __gnu_parallel::__search_template (__RAIter1 *__begin1,* __RAIter1 *__end1,* __RAIter2 *__begin2,* __RAIter2 *__end2,* _Pred *__pred*)

Parallel std::search.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__pred	Find predicate.

Returns

Place of finding in first sequences.

Definition at line 81 of file search.h.

References __calc_borders(), __equally_split(), _GLIBCXX_CALL, and std::min().

3.6.4.54 template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::__sequential_multiway_merge (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _RAIter3 *__target,* const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & *__sentinel,* _DifferenceTp *__length,* _Compare *__comp*)

Sequential multi-way merging switch.

The _GLIBCXX_PARALLEL_DECISION is based on the branching factor and runtime settings.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, possibly larger than the number of elements available.
__sentinel	The sequences have __a __sentinel element.

Returns

End iterator of output sequence.

Definition at line 920 of file multiway_merge.h.

References __is_sorted(), __merge_advance(), _GLIBCXX_CALL, and _GLIBCXX_PARALLEL_LENGTH.

Referenced by multiway_merge(), and multiway_merge_sentinels().

3.6.4.55 template<typename _RAIter , typename _RandomNumberGenerator > void __gnu_parallel::__sequential_random_shuffle (_RAIter *__begin,* _RAIter *__end,* _RandomNumberGenerator & *__rng*)

Sequential cache-efficient random shuffle.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__rng	Random number generator to use.

Definition at line 410 of file random_shuffle.h.

References __random_number_pow2(), __rd_log2(), __round_up_to_pow2(), __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::L2_cache_size, std::min(), std::partial_sum(), and __gnu_parallel::_Settings::TLB_size.

Referenced by __parallel_random_shuffle_drs().

3.6.4.56 template<typename _IIter > void __gnu_parallel::__shrink (std::vector< _IIter > & *__os_starts,* size_t & *__count_to_two,* size_t & *__range_length*)

Combines two ranges into one and thus halves the number of ranges.

Parameters

__os_starts	Start positions worked on (oversampled).
__count_to_two	Counts up to 2.
__range_length	Current length of a chunk.

Definition at line 70 of file list_partition.h.

References std::vector< _Tp, _Alloc >::size().

Referenced by __shrink_and_double().

3.6.4.57 template<typename _IIter > void __gnu_parallel::__shrink_and_double (std::vector< _IIter > & *__os_starts,* size_t & *__count_to_two,* size_t & *__range_length,* const bool *__make_twice*)

Shrinks and doubles the ranges.

Parameters

__os_starts	Start positions worked on (oversampled).
__count_to_two	Counts up to 2.
__range_length	Current length of a chunk.
__make_twice	Whether the `__os_starts` is allowed to be grown or not

Definition at line 50 of file list_partition.h.

References __shrink(), std::vector< _Tp, _Alloc >::resize(), and std::vector< _Tp, _Alloc >::size().

Referenced by list_partition().

3.6.4.58 void __gnu_parallel::__yield ( ) `[inline]`

Yield control to another thread, without waiting for the end of the time slice.

Definition at line 121 of file parallel/compatibility.h.

Referenced by __for_each_template_random_access_workstealing(), and __qsb_local_sort_with_helping().

3.6.4.59 template<typename _IIter , typename _FunctorType > size_t __gnu_parallel::list_partition (const _IIter *__begin,* const _IIter *__end,* _IIter * *__starts,* size_t * *__lengths,* const int *__num_parts,* _FunctorType & *__f,* int *__oversampling* = `0`)

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.

__starts	Start iterators for the resulting parts, dimension `__num_parts+1`. For convenience, `__starts` [__num_parts] contains the end iterator of the sequence.
__lengths	Length of the resulting parts.
__num_parts	Number of parts to split the sequence into.
__f	Functor to be applied to each element by traversing __it
__oversampling	Oversampling factor. If 0, then the partitions will differ in at most {{__end} - {__begin}} __elements. Otherwise, the ratio between the longest and the shortest part is bounded by 1/({__oversampling}{num})

Returns

Length of the whole sequence.

Definition at line 101 of file list_partition.h.

References __shrink_and_double(), and std::vector< _Tp, _Alloc >::size().

3.6.4.60 template<typename _Tp > const _Tp& __gnu_parallel::max (const _Tp & *__a,* const _Tp & *__b*) `[inline]`

Equivalent to std::max.

Definition at line 150 of file parallel/base.h.

3.6.4.61 template<typename _Tp > const _Tp& __gnu_parallel::min (const _Tp & *__a,* const _Tp & *__b*) `[inline]`

Equivalent to std::min.

Definition at line 144 of file parallel/base.h.

Referenced by __for_each_template_random_access_workstealing().

3.6.4.62 template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare > void __gnu_parallel::multiseq_partition (_RanSeqs *__begin_seqs,* _RanSeqs *__end_seqs,* _RankType *__rank,* _RankIterator *__begin_offsets,* _Compare *__comp* = std::less< `typename std::iterator_traits<typename std::iterator_traits<_RanSeqs>::value_type:: first_type>::value_type>()`)

Splits several sorted sequences at a certain global __rank, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the __left side will be chosen from sequences with smaller number.

Parameters

__begin_seqs	Begin of the sequence of iterator pairs.
__end_seqs	End of the sequence of iterator pairs.
__rank	The global rank to partition at.
__begin_offsets	A random-access __sequence __begin where the __result will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective __sequence.
__comp	The ordering functor, defaults to std::less<_Tp>.

Definition at line 122 of file multiseq_selection.h.

References __rd_log2(), _GLIBCXX_CALL, std::vector< _Tp, _Alloc >::begin(), std::distance(), std::priority_queue< _Tp, _Sequence, _Compare >::empty(), std::vector< _Tp, _Alloc >::end(), std::make_pair(), std::max(), std::min(), std::priority_queue< _Tp, _Sequence, _Compare >::pop(), std::priority_queue< _Tp, _Sequence, _Compare >::push(), std::vector< _Tp, _Alloc >::push_back(), and std::priority_queue< _Tp, _Sequence, _Compare >::top().

Referenced by multiway_merge_exact_splitting().

3.6.4.63 template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare > _Tp __gnu_parallel::multiseq_selection (_RanSeqs *__begin_seqs,* _RanSeqs *__end_seqs,* _RankType *__rank,* _RankType & *__offset,* _Compare *__comp =* std::less< _Tp >())

Selects the element at a certain global __rank from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

Parameters

__begin_seqs	Begin of the sequence of iterator pairs.
__end_seqs	End of the sequence of iterator pairs.
__rank	The global rank to partition at.
__offset	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
__comp	The ordering functor, defaults to std::less.

Definition at line 388 of file multiseq_selection.h.

References __rd_log2(), __round_up_to_pow2(), _GLIBCXX_CALL, std::vector< _Tp, _Alloc >::begin(), std::distance(), std::priority_queue< _Tp, _Sequence, _Compare >::empty(), std::vector< _Tp, _Alloc >::end(), std::make_pair(), std::max(), std::min(), std::priority_queue< _Tp, _Sequence, _Compare >::pop(), std::priority_queue< _Tp, _Sequence, _Compare >::push(), std::vector< _Tp, _Alloc >::push_back(), and std::priority_queue< _Tp, _Sequence, _Compare >::top().

3.6.4.64 template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare > _RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator *__seqs_begin,* _RAIterPairIterator *__seqs_end,* _RAIterOut *__target,* _DifferenceTp *__length,* _Compare *__comp,* __gnu_parallel::sequential_tag)

Multiway Merge Frontend.

Merge the sequences specified by seqs_begin and __seqs_end into __target. __seqs_begin and __seqs_end must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the _M_end of the same sequence in their second entry.

Ties are broken arbitrarily. See stable_multiway_merge for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
  for (int __j = 0; __i < 10; ++__j)
    sequences[__i][__j] = __j;
```

```
int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
  { seqs.push(std::make_pair<int*>(sequences[__i],
                                   sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

See Also

stable_multiway_merge

Precondition

All input sequences must be sorted.
Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

Postcondition

[__target, return __value) contains merged __elements from the input sequences.
return __value - __target = min(__length, number of elements in all sequences).

Template Parameters

_RAIterPairIterator	iterator over sequence of pairs of iterators
_RAIterOut	iterator over target sequence
_DifferenceTp	difference type for the sequence
_Compare	strict weak ordering type to compare elements in sequences

Parameters

__seqs_begin	__begin of sequence __sequence
__seqs_end	_M_end of sequence __sequence
__target	target sequence to merge to.
__comp	strict weak ordering to use for element comparison.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

_M_end iterator of output sequence

Definition at line 1418 of file multiway_merge.h.

References __sequential_multiway_merge(), and _GLIBCXX_CALL.

3.6.4.65 template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _RAIter3 *__target,* _DifferenceTp *__length,* _Compare *__comp*)

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 241 of file multiway_merge.h.

References _GLIBCXX_CALL.

3.6.4.66 template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _RAIter3 *__target,* _DifferenceTp *__length,* _Compare *__comp*)

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 360 of file multiway_merge.h.

References _GLIBCXX_CALL.

3.6.4.67 template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType > void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _DifferenceType *__length,* _DifferenceType *__total_length,* _Compare *__comp,* std::vector< std::pair< _DifferenceType, _DifferenceType > > * *__pieces*)

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1120 of file multiway_merge.h.

References __equally_split(), _GLIBCXX_PARALLEL_LENGTH, std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), multiseq_partition(), and std::vector< _Tp, _Alloc >::resize().

Referenced by __parallel_merge_advance().

3.6.4.68 template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _RAIter3 *__target,* _DifferenceTp *__length,* _Compare *__comp*)

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a LoserTree class as selected by `_LT`.

Stability is selected through the used LoserTree class `_LT`.

At least one non-empty sequence is required.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 491 of file multiway_merge.h.

References _GLIBCXX_CALL, and _GLIBCXX_PARALLEL_LENGTH.

3.6.4.69 template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _RAIter3 *__target,* const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & *__sentinel,* _DifferenceTp *__length,* _Compare *__comp*)

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

Template Parameters

UnguardedLoserTree	_Loser Tree variant to use for the unguarded merging.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 662 of file multiway_merge.h.

References __is_sorted(), and _GLIBCXX_CALL.

3.6.4.70 template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare > _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator *__seqs_begin*, _RAIterIterator *__seqs_end*, _RAIter3 *__target*, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type & *__sentinel*, _DifferenceTp *__length*, _Compare *__comp*)

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the LoserTree class `_LT`.

Stability is selected by the used LoserTrees.

Precondition

No input will run out of elements during the merge.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

Definition at line 574 of file multiway_merge.h.

References _GLIBCXX_CALL.

3.6.4.71 template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType > void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator *__seqs_begin*, _RAIterIterator *__seqs_end*, _DifferenceType *__length*, _DifferenceType *__total_length*, _Compare *__comp*, std::vector< std::pair< _DifferenceType, _DifferenceType > > ∗ *__pieces*)

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1035 of file multiway_merge.h.

References _GLIBCXX_PARALLEL_LENGTH, __gnu_parallel::_Settings::get(), and __gnu_parallel::_Settings::merge_oversampling.

3.6.4.72 template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare > _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator *__seqs_begin,* _RAIterPairIterator *__seqs_end,* _RAIterOut *__target,* _DifferenceTp *__length,* _Compare *__comp,* __gnu_parallel::sequential_tag)

Multiway Merge Frontend.

Merge the sequences specified by seqs_begin and __seqs_end into __target. __seqs_begin and __seqs_end must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the _M_end of the same sequence in their second entry.

Ties are broken arbitrarily. See stable_multiway_merge for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the _M_end iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
  for (int __j = 0; __i < 11; ++__j)
    sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
  { seqs.push(std::make_pair<int*>(sequences[__i],
                                   sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

Precondition

All input sequences must be sorted.
Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.
For each `__i`, `__seqs_begin`[__i].second must be the end marker of the sequence, but also reference the one more __sentinel element.

Postcondition

[__target, return __value) contains merged __elements from the input sequences.
return __value - __target = min(__length, number of elements in all sequences).

See Also

stable_multiway_merge_sentinels

Template Parameters

_RAIterPairIterator	iterator over sequence of pairs of iterators
_RAIterOut	iterator over target sequence
_DifferenceTp	difference type for the sequence
_Compare	strict weak ordering type to compare elements in sequences

Parameters

__seqs_begin	__begin of sequence __sequence
__seqs_end	_M_end of sequence __sequence
__target	target sequence to merge to.
__comp	strict weak ordering to use for element comparison.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

_M_end iterator of output sequence

Definition at line 1782 of file multiway_merge.h.

References __sequential_multiway_merge(), and _GLIBCXX_CALL.

3.6.4.73 template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Splitter , typename _Compare > _RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator *__seqs_begin,* _RAIterIterator *__seqs_end,* _RAIter3 *__target,* _Splitter *__splitter,* _DifferenceTp *__length,* _Compare *__comp,* _ThreadIndex *__num_threads*)

Parallel multi-way merge routine.

The _GLIBCXX_PARALLEL_DECISION is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

Template Parameters

_Splitter	functor to split input (either __exact or sampling based)
__stable	Stable merging incurs a performance penalty.
__sentinel	Ignored.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

End iterator of output sequence.

Definition at line 1225 of file multiway_merge.h.

References __is_sorted(), _GLIBCXX_CALL, _GLIBCXX_PARALLEL_LENGTH, __gnu_parallel::_Settings::get(), std::make_pair(), and __gnu_parallel::_Settings::merge_oversampling.

Referenced by __parallel_merge_advance().

3.6.4.74 template<bool __stable, bool __exact, typename _RAIter , typename _Compare > void __gnu_parallel::parallel_sort_mwms (_RAIter *__begin*, _RAIter *__end*, _Compare *__comp*, _ThreadIndex *__num_threads*)

PMWMS main call.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.
__num_threads	Number of threads to use.

Definition at line 395 of file multiway_mergesort.h.

References _GLIBCXX_CALL, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_offsets, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary, __gnu_parallel::_Settings::get(), std::vector< _Tp, _Alloc >::resize(), and __gnu_parallel::_Settings::sort_mwms_oversampling.

3.6.4.75 template<bool __stable, bool __exact, typename _RAIter , typename _Compare > void __gnu_parallel::parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > * *__sd*, _Compare & *__comp*)

PMWMS code executed by each thread.

Parameters

__sd	Pointer to algorithm data.
__comp	Comparator.

Definition at line 308 of file multiway_mergesort.h.

References __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary, __gnu_parallel::_Settings::get(), std::make_pair(), __gnu_parallel::_Settings::sort_mwms_oversampling, and std::uninitialized_copy().

3.6.5 Variable Documentation

3.6.5.1 const int __gnu_parallel::_CASable_bits `[static]`

Number of bits of _CASable.

Definition at line 130 of file types.h.

Referenced by __decode2(), and __encode2().

3.6.5.2 const _CASable __gnu_parallel::_CASable_mask [static]

_CASable with the right half of bits set to 1.

Definition at line 133 of file types.h.

Referenced by __decode2().

3.7 __gnu_pbds Namespace Reference

Classes

- struct associative_tag

  Basic associative-container.
- class basic_branch
- struct basic_branch_tag

  Basic branch structure.
- class basic_hash_table
- struct basic_hash_tag

  Basic hash structure.
- struct basic_invalidation_guarantee
- struct binary_heap_tag

  Binary-heap (array-based).
- struct binomial_heap_tag

  Binomial-heap.
- class cc_hash_max_collision_check_resize_trigger

  A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.
- class cc_hash_table
- struct cc_hash_tag

  Collision-chaining hash.
- struct container_error

  Base class for exceptions.
- struct container_tag

  Base data structure tag.
- struct container_traits

  Container traits.
- struct container_traits_base< binary_heap_tag >

  Specialization, binary heap.
- struct container_traits_base< binomial_heap_tag >

  Specialization, binomial heap.
- struct container_traits_base< cc_hash_tag >

  Specialization, cc hash.
- struct container_traits_base< gp_hash_tag >

  Specialization, gp hash.
- struct container_traits_base< list_update_tag >

  Specialization, list update.
- struct container_traits_base< ov_tree_tag >

Specialization, ov tree.

- struct container_traits_base< pairing_heap_tag >

  Specialization, pairing heap.
- struct container_traits_base< pat_trie_tag >

  Specialization, pat trie.
- struct container_traits_base< rb_tree_tag >

  Specialization, rb tree.
- struct container_traits_base< rc_binomial_heap_tag >

  Specialization, rc binomial heap.
- struct container_traits_base< splay_tree_tag >

  Specialization, splay tree.
- struct container_traits_base< thin_heap_tag >

  Specialization, thin heap.
- class direct_mask_range_hashing

  A mask range-hashing class (uses a bitmask).
- class direct_mod_range_hashing

  A mod range-hashing class (uses the modulo function).
- class gp_hash_table
- struct gp_hash_tag

  General-probing hash.
- class hash_exponential_size_policy

  A size policy whose sequence of sizes form an exponential sequence (typically powers of 2.
- class hash_load_check_resize_trigger

  A resize trigger policy based on a load check. It keeps the load factor between some load factors load_min and load_max.
- class hash_prime_size_policy

  A size policy whose sequence of sizes form a nearly-exponential sequence of primes.
- class hash_standard_resize_policy

  A resize policy which delegates operations to size and trigger policies.
- struct insert_error

  An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unabvailable, in which case the allocator_type's exception will be thrown).
- struct join_error

  A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.
- class linear_probe_fn

  A probe sequence policy using fixed increments.
- class list_update
- struct list_update_tag

  List-update.
- class lu_counter_policy
- class lu_move_to_front_policy
- struct null_node_update

  A null node updator, indicating that no node updates are required.
- struct null_type

  Represents no type, or absence of type, for template tricks.
- struct ov_tree_tag

  Ordered-vector tree.
- struct pairing_heap_tag

  Pairing-heap.
- struct pat_trie_tag

  PATRICIA trie.
- struct point_invalidation_guarantee
- class priority_queue
- struct priority_queue_tag

  Basic priority-queue.
- class quadratic_probe_fn

  A probe sequence policy using square increments.
- struct range_invalidation_guarantee
- struct rb_tree_tag

  Red-black tree.
- struct rc_binomial_heap_tag

  Redundant-counter binomial-heap.
- struct resize_error

  A container cannot be resized.
- class sample_probe_fn

  A sample probe policy.
- class sample_range_hashing

  A sample range-hashing functor.
- class sample_ranged_hash_fn

  A sample ranged-hash functor.
- class sample_ranged_probe_fn

  A sample ranged-probe functor.
- class sample_resize_policy

  A sample resize policy.
- class sample_resize_trigger

  A sample resize trigger policy.
- class sample_size_policy

  A sample size policy.
- class sample_tree_node_update

  A sample node updator.
- struct sample_trie_access_traits

  A sample trie element access traits.
- class sample_trie_node_update

  A sample node updator.
- struct sample_update_policy

  A sample list-update policy.
- struct sequence_tag

  Basic sequence.
- struct splay_tree_tag

  Splay tree.
- struct string_tag

  Basic string container, inclusive of strings, ropes, etc.
- struct thin_heap_tag

  Thin heap.
- class tree

- class tree_order_statistics_node_update

  Functor updating ranks of entrees.
- struct tree_tag

  Basic tree structure.
- class trie
- class trie_order_statistics_node_update

  Functor updating ranks of entrees.
- class trie_prefix_search_node_update

  A node updator that allows tries to be searched for the range of values that match a certain prefix.
- struct trie_string_access_traits
- struct trie_tag

  Basic trie structure.
- struct trivial_iterator_tag

  A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.

Typedefs

- typedef void trivial_iterator_difference_type

Functions

- void **__throw_container_error** ()
- void **__throw_insert_error** ()
- void **__throw_join_error** ()
- void **__throw_resize_error** ()

3.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

3.8 __gnu_profile Namespace Reference

Classes

- class __container_size_info

  A container size instrumentation line in the object table.
- class __container_size_stack_info

  A container size instrumentation line in the stack table.
- class __hashfunc_info

  A hash performance instrumentation line in the object table.
- class __hashfunc_stack_info

  A hash performance instrumentation line in the stack table.
- class __list2vector_info

  A list-to-vector instrumentation line in the object table.
- class __map2umap_info

  A map-to-unordered_map instrumentation line in the object table.
- class __map2umap_stack_info

A map-to-unordered_map instrumentation line in the stack table.
- class __object_info_base

  Base class for a line in the object table.
- struct __reentrance_guard

  Reentrance guard.
- class __stack_hash

  Hash function for summary trace using call stack as index.
- class __stack_info_base

  Base class for a line in the stack table.
- class __trace_base

  Base class for all trace producers.
- class __trace_container_size

  Container size instrumentation trace producer.
- class __trace_hash_func

  Hash performance instrumentation producer.
- class __trace_hashtable_size

  Hashtable size instrumentation trace producer.
- class __trace_map2umap

  Map-to-unordered_map instrumentation producer.
- class __trace_vector_size

  Hashtable size instrumentation trace producer.
- class __trace_vector_to_list

  Vector-to-list instrumentation producer.
- class __vector2list_info

  A vector-to-list instrumentation line in the object table.
- class __vector2list_stack_info

  A vector-to-list instrumentation line in the stack table.
- struct __warning_data

  Representation of a warning.

Typedefs

- typedef std::::vector
  < __cost_factor ∗ > **__cost_factor_vector**
- typedef std::::unordered_map
  < std::string, std::string > __env_t
- typedef void ∗ **__instruction_address_t**
- typedef const void ∗ **__object_t**
- typedef std::::vector
  < __instruction_address_t > **__stack_npt**
- typedef __stack_npt ∗ **__stack_t**
- typedef std::::vector
  < __warning_data > **__warning_vector_t**

Enumerations

- enum **__state_type** { **__ON**, **__OFF**, **__INVALID** }

Functions

- std::size_t **__env_to_size_t** (const char ∗__env_var, std::size_t __default_value)
- template<typename _InputIterator , typename _Function >
  _Function **__for_each** (_InputIterator __first, _InputIterator __last, _Function __f)
- __stack_t **__get_stack** ()
- template<typename _Container >
  void **__insert_top_n** (_Container &__output, const typename _Container::value_type &__value, typename _Container::size_type __n)
- bool **__is_invalid** ()
- bool **__is_off** ()
- bool **__is_on** ()
- int **__log2** (std::size_t __size)
- int **__log_magnitude** (float __f)
- float **__map_erase_cost** (std::size_t __size)
- float **__map_find_cost** (std::size_t __size)
- float **__map_insert_cost** (std::size_t __size)
- std::size_t **__max_mem** ()
- FILE ∗ **__open_output_file** (const char ∗__extension)
- bool __profcxx_init ()
- void **__profcxx_init_unconditional** ()
- void **__read_cost_factors** ()
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator **__remove** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- void __report (void)
- void **__set_cost_factors** ()
- void **__set_max_mem** ()
- void **__set_max_stack_trace_depth** ()
- void **__set_max_warn_count** ()
- void **__set_trace_path** ()
- std::size_t **__size** (__stack_t __stack)
- std::size_t **__stack_max_depth** ()
- template<typename _Container >
  void **__top_n** (const _Container &__input, _Container &__output, typename _Container::size_type __n)
- void **__trace_hash_func_construct** (const void ∗)
- void **__trace_hash_func_destruct** (const void ∗, std::size_t, std::size_t, std::size_t)
- void **__trace_hash_func_init** ()
- void **__trace_hash_func_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__trace_hashtable_size_construct** (const void ∗, std::size_t)
- void **__trace_hashtable_size_destruct** (const void ∗, std::size_t, std::size_t)
- void **__trace_hashtable_size_init** ()
- void **__trace_hashtable_size_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__trace_hashtable_size_resize** (const void ∗, std::size_t, std::size_t)
- void **__trace_list_to_set_construct** (const void ∗)
- void **__trace_list_to_set_destruct** (const void ∗)
- void **__trace_list_to_set_find** (const void ∗, std::size_t)
- void **__trace_list_to_set_insert** (const void ∗, std::size_t, std::size_t)
- void **__trace_list_to_set_invalid_operator** (const void ∗)
- void **__trace_list_to_set_iterate** (const void ∗, std::size_t)
- void **__trace_list_to_slist_construct** (const void ∗)
- void **__trace_list_to_slist_destruct** (const void ∗)

- void **__trace_list_to_slist_init** ()
- void **__trace_list_to_slist_operation** (const void ∗)
- void **__trace_list_to_slist_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__trace_list_to_slist_rewind** (const void ∗)
- void **__trace_list_to_vector_construct** (const void ∗)
- void **__trace_list_to_vector_destruct** (const void ∗)
- void **__trace_list_to_vector_init** ()
- void **__trace_list_to_vector_insert** (const void ∗, std::size_t, std::size_t)
- void **__trace_list_to_vector_invalid_operator** (const void ∗)
- void **__trace_list_to_vector_iterate** (const void ∗, std::size_t)
- void **__trace_list_to_vector_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__trace_list_to_vector_resize** (const void ∗, std::size_t, std::size_t)
- void **__trace_map_to_unordered_map_construct** (const void ∗)
- void **__trace_map_to_unordered_map_destruct** (const void ∗)
- void **__trace_map_to_unordered_map_erase** (const void ∗, std::size_t, std::size_t)
- void **__trace_map_to_unordered_map_find** (const void ∗, std::size_t)
- void **__trace_map_to_unordered_map_init** ()
- void **__trace_map_to_unordered_map_insert** (const void ∗, std::size_t, std::size_t)
- void **__trace_map_to_unordered_map_invalidate** (const void ∗)
- void **__trace_map_to_unordered_map_iterate** (const void ∗, std::size_t)
- void **__trace_map_to_unordered_map_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__trace_vector_size_construct** (const void ∗, std::size_t)
- void **__trace_vector_size_destruct** (const void ∗, std::size_t, std::size_t)
- void **__trace_vector_size_init** ()
- void **__trace_vector_size_report** (FILE ∗, __warning_vector_t &)
- void **__trace_vector_size_resize** (const void ∗, std::size_t, std::size_t)
- void **__trace_vector_to_list_construct** (const void ∗)
- void **__trace_vector_to_list_destruct** (const void ∗)
- void **__trace_vector_to_list_find** (const void ∗, std::size_t)
- void **__trace_vector_to_list_init** ()
- void **__trace_vector_to_list_insert** (const void ∗, std::size_t, std::size_t)
- void **__trace_vector_to_list_invalid_operator** (const void ∗)
- void **__trace_vector_to_list_iterate** (const void ∗, std::size_t)
- void **__trace_vector_to_list_report** (FILE ∗, __warning_vector_t &)
- void **__trace_vector_to_list_resize** (const void ∗, std::size_t, std::size_t)
- bool **__turn** (__state_type __s)
- bool **__turn_off** ()
- bool **__turn_on** ()
- void **__write** (FILE ∗__f, __stack_t __stack)
- void **__write_cost_factors** ()
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__state_type, __state, __INVALID)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_hash_func ∗, _S_hash_func, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_hashtable_size ∗, _S_hashtable_size, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_map2umap ∗, _S_map2umap, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_vector_size ∗, _S_vector_size, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_vector_to_list ∗, _S_vector_to_list, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_list_to_slist ∗, _S_list_to_slist, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_list_to_vector ∗, _S_list_to_vector, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_shift_cost_factor,{"__vector_shift_cost_factor", 1.0})

- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_iterate_cost_factor,{"__vector_iterate_cost_factor", 1.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_resize_cost_factor,{"__vector_resize_cost_factor", 1.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __list_shift_cost_factor,{"__list_shift_cost_factor", 0.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __list_iterate_cost_factor,{"__list_iterate_cost_factor", 10.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __list_resize_cost_factor,{"__list_resize_cost_factor", 0.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_insert_cost_factor,{"__map_insert_cost_factor", 1.5})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_erase_cost_factor,{"__map_erase_cost_factor", 1.5})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_find_cost_factor,{"__map_find_cost_factor", 1})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_iterate_cost_factor,{"__map_iterate_cost_factor", 2.3})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_insert_cost_factor,{"__umap_insert_cost_factor", 12.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_erase_cost_factor,{"__umap_erase_cost_factor", 12.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_find_cost_factor,{"__umap_find_cost_factor", 10.0})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_iterate_cost_factor,{"__umap_iterate_cost_factor", 1.7})
- **_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor_vector ∗, __cost_factors, 0)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (const char ∗, _S_trace_file_name, _GLIBCXX_PROFILE_TRACE_PATH_ROOT)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (std::size_t, _S_max_warn_count, _GLIBCXX_PROFILE_MAX_WARN_COUNT)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (std::size_t, _S_max_stack_depth, _GLIBCXX_PROFILE_MAX_STACK_DEPTH)
- **_GLIBCXX_PROFILE_DEFINE_DATA** (std::size_t, _S_max_mem, _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC)
- **_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA** (__env_t, __env)
- _GLIBCXX_PROFILE_DEFINE_UNINIT_DATA (__gnu_cxx::__mutex, __global_lock)

3.8.1 Detailed Description

GNU profile code for public use.

3.8.2 Typedef Documentation

3.8.2.1 typedef std:: ::unordered_map<std::string, std::string> __gnu_profile::__env_t

Internal environment. Values can be set one of two ways: 1. In config file "var = value". The default config file path is libstdcxx-profile.conf. 2. By setting process environment variables. For instance, in a Bash shell you can set the unit cost of iterating through a map like this: export __map_iterate_cost_factor=5.0. If a value is set both in the input file and through an environment variable, the environment value takes precedence.

Definition at line 65 of file profiler_trace.h.

3.8.3 Function Documentation

3.8.3.1 bool __gnu_profile::__profcxx_init ( ) [inline]

This function must be called by each instrumentation point.

The common path is inlined fully.

Definition at line 649 of file profiler_trace.h.

3.8.3.2 void __gnu_profile::__report (void) [inline]

Final report method, registered with **atexit**.

This can also be called directly by user code, including signal handlers. It is protected against deadlocks by the reentrance guard in profiler.h. However, when called from a signal handler that triggers while within __gnu_profile (under the guarded zone), no output will be produced.

Definition at line 440 of file profiler_trace.h.

References std::min().

3.8.3.3 __gnu_profile::_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA (__gnu_cxx::__mutex , __global_lock)

Master lock.

3.9 __gnu_sequential Namespace Reference

3.9.1 Detailed Description

GNU sequential classes for public use.

3.10 abi Namespace Reference

3.10.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to __cxxabiv1, but user programs should use the alias 'abi'. A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at `http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8`).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at `http://www.codesourcery.com/cxx-abi/` along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in <cxxabi.h>, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

3.11 std Namespace Reference

Namespaces

- namespace __debug
- namespace __detail
- namespace __parallel

- namespace __profile
- namespace chrono
- namespace decimal
- namespace placeholders
- namespace regex_constants
- namespace rel_ops
- namespace this_thread
- namespace tr1
- namespace tr2

Classes

- struct __atomic_base

  Base class for atomic integrals.
- struct __atomic_base< _PTp * >

  Partial specialization for pointer types.
- struct __atomic_flag_base

  Base type for atomic_flag.
- class __codecvt_abstract_base

  Common base for codecvt functions.
- class __ctype_abstract_base

  Common base for ctype facet.
- class __has_iterator_category_helper

  Traits class for iterators.
- struct __is_location_invariant
- struct __numeric_limits_base

  Part of std::numeric_limits.
- struct _Base_bitset
- struct _Base_bitset< 0 >
- struct _Base_bitset< 1 >
- class _Deque_base
- struct _Deque_iterator

  A deque::iterator.
- struct _Derives_from_binary_function

  Determines if the type _Tp derives from binary_function.
- struct _Derives_from_unary_function

  Determines if the type _Tp derives from unary_function.
- class _Function_base

  Base class of all polymorphic function object wrappers.
- struct _Fwd_list_base

  Base class for forward_list.
- struct _Fwd_list_const_iterator

  A forward_list::const_iterator.
- struct _Fwd_list_iterator

  A forward_list::iterator.
- struct _Fwd_list_node

  A helper node class for forward_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

- struct _Fwd_list_node_base

  A helper basic node class for forward_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.
- class _Hashtable
- class _List_base

  See bits/stl_deque.h's _Deque_base for an explanation.
- struct _List_const_iterator

  A list::const_iterator.
- struct _List_iterator

  A list::iterator.
- struct _List_node

  An actual node in the list.
- struct _Maybe_get_result_type

  If we have found a result_type, extract it.
- struct _Maybe_unary_or_binary_function
- struct _Maybe_unary_or_binary_function< _Res, _T1 >

  Derives from `unary_function`, as appropriate.
- struct _Maybe_unary_or_binary_function< _Res, _T1, _T2 >

  Derives from `binary_function`, as appropriate.
- struct _Maybe_wrap_member_pointer
- struct _Maybe_wrap_member_pointer< _Tp _Class::∗ >
- class _Mem_fn< _Res(_Class::∗)(_ArgTypes...) const >

  Implementation of `mem_fn` for const member function pointers.
- class _Mem_fn< _Res(_Class::∗)(_ArgTypes...) const volatile >

  Implementation of `mem_fn` for const volatile member function pointers.
- class _Mem_fn< _Res(_Class::∗)(_ArgTypes...) volatile >

  Implementation of `mem_fn` for volatile member function pointers.
- class _Mem_fn< _Res(_Class::∗)(_ArgTypes...)>

  Implementation of `mem_fn` for member function pointers.
- class _Mu< _Arg, false, false >
- class _Mu< _Arg, false, true >
- class _Mu< _Arg, true, false >
- class _Mu< reference_wrapper< _Tp >, false, false >
- struct _Placeholder

  The type of placeholder objects defined by libstdc++.
- struct _Reference_wrapper_base
- struct _Safe_tuple_element
- struct _Safe_tuple_element_impl
- struct _Safe_tuple_element_impl< __i, _Tuple, false >
- class _Temporary_buffer
- struct _Tuple_impl< _Idx >
- struct _Tuple_impl< _Idx, _Head, _Tail...>
- struct _Vector_base

  See bits/stl_deque.h's _Deque_base for an explanation.
- struct _Weak_result_type
- struct _Weak_result_type_impl
- struct _Weak_result_type_impl< _Res(&)(_ArgTypes...)>

  Retrieve the result type for a function reference.

- struct _Weak_result_type_impl< _Res(∗)(_ArgTypes...)>

  Retrieve the result type for a function pointer.
- struct _Weak_result_type_impl< _Res(_ArgTypes...)>

  Retrieve the result type for a function type.
- struct _Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) const >

  Retrieve result type for a const member function pointer.
- struct _Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) const volatile >

  Retrieve result type for a const volatile member function pointer.
- struct _Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) volatile >

  Retrieve result type for a volatile member function pointer.
- struct _Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...)>

  Retrieve result type for a member function pointer.
- struct add_const

  add_const
- struct add_cv

  add_cv
- struct add_lvalue_reference

  add_lvalue_reference
- struct add_pointer

  add_pointer
- struct add_rvalue_reference

  add_rvalue_reference
- struct add_volatile

  add_volatile
- struct adopt_lock_t

  Assume the calling thread has already obtained mutex ownership and manage it.
- struct aligned_storage

  Alignment type.
- struct alignment_of

  alignment_of
- struct allocator

  The standard allocator, as per [20.4].
- class allocator< void >

  allocator<void> specialization.
- struct allocator_arg_t

  [allocator.tag]
- struct allocator_traits

  Uniform interface to all allocator types.
- struct array

  A standard container for storing a fixed size sequence of elements.
- struct atomic

  Generic atomic type, primary class template.
- struct atomic< _Tp ∗ >

  Partial specialization for pointer types.
- struct atomic< bool >

  Explicit specialization for bool.
- struct atomic< char >

Explicit specialization for char.

- struct atomic< char16_t >

  Explicit specialization for char16_t.
- struct atomic< char32_t >

  Explicit specialization for char32_t.
- struct atomic< int >

  Explicit specialization for int.
- struct atomic< long >

  Explicit specialization for long.
- struct atomic< long long >

  Explicit specialization for long long.
- struct atomic< short >

  Explicit specialization for short.
- struct atomic< signed char >

  Explicit specialization for signed char.
- struct atomic< unsigned char >

  Explicit specialization for unsigned char.
- struct atomic< unsigned int >

  Explicit specialization for unsigned int.
- struct atomic< unsigned long >

  Explicit specialization for unsigned long.
- struct atomic< unsigned long long >

  Explicit specialization for unsigned long long.
- struct atomic< unsigned short >

  Explicit specialization for unsigned short.
- struct atomic< wchar_t >

  Explicit specialization for wchar_t.
- struct atomic_bool

  atomic_bool
- struct atomic_flag

  atomic_flag
- class auto_ptr

  A simple smart pointer providing strict ownership semantics.
- struct auto_ptr_ref
- class back_insert_iterator

  Turns assignment into insertion.
- class bad_alloc

  Exception possibly thrown by `new`.
  `bad_alloc` *(or classes derived from it) is used to report allocation errors from the throwing forms of* `new`*.*
- class bad_cast

  Thrown during incorrect typecasting.
  If you attempt an invalid `dynamic_cast` *expression, an instance of this class (or something derived from this class) is thrown.*
- class bad_exception
- class bad_function_call

  Exception class thrown when class template function's operator() is called with an empty target.
- class bad_typeid

  Thrown when a NULL pointer in a `typeid` *expression is used.*

- class bad_weak_ptr

    Exception possibly thrown by `shared_ptr`.
- class basic_filebuf

    The actual work of input and output (for files).
- class basic_fstream

    Controlling input and output for files.
- class basic_ifstream

    Controlling input for files.
- class basic_ios

    Template class basic_ios, virtual base class for all stream classes.
- class basic_iostream

    Template class basic_iostream.
- class basic_istream

    Template class basic_istream.
- class basic_istringstream

    Controlling input for std::string.
- class basic_ofstream

    Controlling output for files.
- class basic_ostream

    Template class basic_ostream.
- class basic_ostringstream

    Controlling output for std::string.
- class basic_regex
- class basic_streambuf

    The actual work of input and output (interface).
- class basic_string

    Managing sequences of characters and character-like objects.
- class basic_stringbuf

    The actual work of input and output (for std::string).
- class basic_stringstream

    Controlling input and output for std::string.
- class bernoulli_distribution

    A Bernoulli random number distribution.
- struct bidirectional_iterator_tag

    Bidirectional iterators support a superset of forward iterator operations.
- struct binary_function
- class binary_negate

    One of the negation functors.
- class binder1st

    One of the binder functors.
- class binder2nd

    One of the binder functors.
- class binomial_distribution

    A discrete binomial random number distribution.
- class cauchy_distribution

    A cauchy_distribution random number distribution.
- struct char_traits

Basis for explicit traits specializations.
- struct char_traits< __gnu_cxx::character< V, I, S > >

  char_traits<__gnu_cxx::character> specialization.
- struct char_traits< char >

  21.1.3.1 char_traits specializations
- struct char_traits< wchar_t >

  21.1.3.2 char_traits specializations
- class chi_squared_distribution

  A chi_squared_distribution random number distribution.
- class codecvt

  Primary class template codecvt.
  NB: Generic, mostly useless implementation.
- class codecvt< _InternT, _ExternT, encoding_state >

  codecvt<InternT, _ExternT, encoding_state> specialization.
- class codecvt< char, char, mbstate_t >

  class codecvt<char, char, mbstate_t> specialization.
- class codecvt< wchar_t, char, mbstate_t >

  class codecvt<wchar_t, char, mbstate_t> specialization.
- class codecvt_base

  Empty base class for codecvt facet [22.2.1.5].
- class codecvt_byname

  class codecvt_byname [22.2.1.6].
- class collate

  Facet for localized string comparison.
- class collate_byname

  class collate_byname [22.2.4.2].
- struct complex
- struct complex< double >

  26.2.3 complex specializations complex<double> specialization
- struct complex< float >

  26.2.3 complex specializations complex<float> specialization
- struct complex< long double >

  26.2.3 complex specializations complex<long double> specialization
- class condition_variable

  condition_variable
- class condition_variable_any

  condition_variable_any
- struct conditional

  Define a member typedef `type` *to one of two argument types.*
- class const_mem_fun1_ref_t

  One of the adaptors for member pointers.
- class const_mem_fun1_t

  One of the adaptors for member pointers.
- class const_mem_fun_ref_t

  One of the adaptors for member pointers.
- class const_mem_fun_t

  One of the adaptors for member pointers.

- class ctype

  Primary class template ctype facet.
  This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.
- class ctype< char >

  The ctype<char> specialization.
  This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.
- class ctype< wchar_t >

  The ctype<wchar_t> specialization.
  This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
- struct ctype_base

  Base class for ctype.
- class ctype_byname

  class ctype_byname [22.2.1.2].
- class ctype_byname< char >

  22.2.1.4 Class ctype_byname specializations.
- class decay

  decay
- struct default_delete

  Primary template, default_delete.
- struct default_delete< _Tp[]>

  Specialization, default_delete.
- struct defer_lock_t

  Do not acquire ownership of the mutex.
- class deque

  A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.
- class discard_block_engine
- class discrete_distribution

  A discrete_distribution random number distribution.
- struct divides

  One of the math functors.
- class domain_error
- struct enable_if

  Define a member typedef `type` only if a boolean constant is true.
- class enable_shared_from_this

  Base class allowing use of member function shared_from_this.
- struct equal_to

  One of the comparison functors.
- class error_category

  error_category
- struct error_code

  error_code
- struct error_condition

  error_condition
- class exception

  Base class for all library exceptions.

- class exponential_distribution

  An exponential continuous distribution for random numbers.
- struct extent

  extent
- class extreme_value_distribution

  A extreme_value_distribution random number distribution.
- class fisher_f_distribution

  A fisher_f_distribution random number distribution.
- struct forward_iterator_tag

  Forward iterators support a superset of input iterator operations.
- class forward_list

  A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class fpos

  Class representing stream positions.
- class front_insert_iterator

  Turns assignment into insertion.
- class function< _Res(_ArgTypes...)>

  Primary class template for std::function.
  Polymorphic function wrapper.
- class future_error

  Exception type thrown by futures.
- class gamma_distribution

  A gamma continuous distribution for random numbers.
- class geometric_distribution

  A discrete geometric random number distribution.
- struct greater

  One of the comparison functors.
- struct greater_equal

  One of the comparison functors.
- class gslice

  Class defining multi-dimensional subset of an array.
- class gslice_array

  Reference to multi-dimensional subset of an array.
- struct has_trivial_copy_assign

  has_trivial_copy_assign (temporary legacy)
- struct has_trivial_copy_constructor

  has_trivial_copy_constructor (temporary legacy)
- struct has_trivial_default_constructor

  has_trivial_default_constructor (temporary legacy)
- struct has_virtual_destructor

  has_virtual_destructor
- struct hash< __debug::bitset< _Nb > >

  std::hash specialization for bitset.
- struct hash< __debug::vector< bool, _Alloc > >

  std::hash specialization for vector<bool>.
- struct hash< __gnu_cxx::__u16vstring >

  std::hash specialization for __u16vstring.

- struct hash< __gnu_cxx::__u32vstring >

    std::hash specialization for __u32vstring.
- struct hash< __gnu_cxx::__vstring >

    std::hash specialization for __vstring.
- struct hash< __gnu_cxx::__wvstring >

    std::hash specialization for __wvstring.
- struct hash< __gnu_cxx::throw_value_limit >

    Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.
- struct hash< __gnu_cxx::throw_value_random >

    Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.
- struct hash< __profile::bitset< _Nb > >

    std::hash specialization for bitset.
- struct hash< __profile::vector< bool, _Alloc > >

    std::hash specialization for vector<bool>.
- struct hash< __shared_ptr< _Tp, _Lp > >

    std::hash specialization for __shared_ptr.
- struct hash< _Tp ∗ >

    Partial specializations for pointer types.
- struct hash< bool >

    Explicit specialization for bool.
- struct hash< char >

    Explicit specialization for char.
- struct hash< char16_t >

    Explicit specialization for char16_t.
- struct hash< char32_t >

    Explicit specialization for char32_t.
- struct hash< double >

    Specialization for double.
- struct hash< error_code >

    std::hash specialization for error_code.
- struct hash< float >

    Specialization for float.
- struct hash< int >

    Explicit specialization for int.
- struct hash< long >

    Explicit specialization for long.
- struct hash< long double >

    Specialization for long double.
- struct hash< long long >

    Explicit specialization for long long.
- struct hash< shared_ptr< _Tp > >

    std::hash specialization for shared_ptr.
- struct hash< short >

    Explicit specialization for short.
- struct hash< signed char >

    Explicit specialization for signed char.
- struct hash< string >

std::hash specialization for string.

- struct hash< thread::id >

  std::hash specialization for thread::id.
- struct hash< type_index >

  std::hash specialization for type_index.
- struct hash< u16string >

  std::hash specialization for u16string.
- struct hash< u32string >

  std::hash specialization for u32string.
- struct hash< unique_ptr< _Tp, _Dp > >

  std::hash specialization for unique_ptr.
- struct hash< unsigned char >

  Explicit specialization for unsigned char.
- struct hash< unsigned int >

  Explicit specialization for unsigned int.
- struct hash< unsigned long >

  Explicit specialization for unsigned long.
- struct hash< unsigned long long >

  Explicit specialization for unsigned long long.
- struct hash< unsigned short >

  Explicit specialization for unsigned short.
- struct hash< wchar_t >

  Explicit specialization for wchar_t.
- struct hash< wstring >

  std::hash specialization for wstring.
- struct hash<::bitset< _Nb > >

  std::hash specialization for bitset.
- struct hash<::vector< bool, _Alloc > >

  std::hash specialization for vector<bool>.
- class independent_bits_engine
- class indirect_array

  Reference to arbitrary subset of an array.
- class initializer_list

  initializer_list
- struct input_iterator_tag

  Marking input iterators.
- class insert_iterator

  Turns assignment into insertion.
- struct integral_constant

  integral_constant
- class invalid_argument
- class ios_base

  The base of the I/O class hierarchy.
  This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` *when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- struct is_abstract

  is_abstract

- struct is_arithmetic

  is_arithmetic
- struct is_array

  is_array
- struct is_assignable

  is_assignable
- struct is_base_of

  is_base_of
- struct is_bind_expression

  Determines if the given type _Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].
- struct is_bind_expression< _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct is_bind_expression< _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct is_bind_expression< const _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct is_bind_expression< const _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct is_bind_expression< const volatile _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct is_bind_expression< volatile _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct is_bind_expression< volatile _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct is_class

  is_class
- struct is_compound

  is_compound
- struct is_const

  is_const
- struct is_constructible

  is_constructible
- struct is_convertible

  is_convertible
- struct is_copy_assignable

  is_copy_assignable
- struct is_copy_constructible

  is_copy_constructible
- struct is_default_constructible

  is_default_constructible
- struct is_destructible

  is_destructible
- struct is_empty

  is_empty

- struct is_enum

  is_enum
- struct is_error_code_enum

  is_error_code_enum
- struct is_error_code_enum< future_errc >

  Specialization.
- struct is_error_condition_enum

  is_error_condition_enum
- struct is_floating_point

  is_floating_point
- struct is_function

  is_function
- struct is_fundamental

  is_fundamental
- struct is_integral

  is_integral
- struct is_literal_type

  is_literal_type
- struct is_lvalue_reference

  is_lvalue_reference
- struct is_member_function_pointer

  is_member_function_pointer
- struct is_member_object_pointer

  is_member_object_pointer
- struct is_member_pointer

  is_member_pointer
- struct is_move_assignable

  is_move_assignable
- struct is_move_constructible

  is_move_constructible
- struct is_nothrow_assignable

  is_nothrow_assignable
- struct is_nothrow_constructible

  is_nothrow_constructible
- struct is_nothrow_copy_assignable

  is_nothrow_copy_assignable
- struct is_nothrow_copy_constructible

  is_nothrow_copy_constructible
- struct is_nothrow_default_constructible

  is_nothrow_default_constructible
- struct is_nothrow_destructible

  is_nothrow_destructible
- struct is_nothrow_move_assignable

  is_nothrow_move_assignable
- struct is_nothrow_move_constructible

  is_nothrow_move_constructible
- struct is_object

is_object
- struct is_placeholder

  Determines if the given type _Tp is a placeholder in a bind() expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct is_placeholder< _Placeholder< _Num > >
- struct is_pod

  is_pod
- struct is_pointer

  is_pointer
- struct is_polymorphic

  is_polymorphic
- struct is_reference

  is_reference
- struct is_rvalue_reference

  is_rvalue_reference
- struct is_same

  is_same
- struct is_scalar

  is_scalar
- struct is_signed

  is_signed
- struct is_standard_layout

  is_standard_layout
- struct is_trivial

  is_trivial
- struct is_trivially_destructible

  is_trivially_constructible (still unimplemented)
- struct is_union

  is_union
- struct is_unsigned

  is_unsigned
- struct is_void

  is_void
- struct is_volatile

  is_volatile
- class istream_iterator

  Provides input iterator semantics for streams.
- class istreambuf_iterator

  Provides input iterator semantics for streambufs.
- struct iterator

  Common iterator class.
- struct iterator_traits< _Tp ∗ >

  Partial specialization for pointer types.
- struct iterator_traits< const _Tp ∗ >

  Partial specialization for const pointer types.
- class length_error
- struct less

  One of the comparison functors.

- struct less_equal

  One of the comparison functors.
- class linear_congruential_engine

  A model of a linear congruential random number generator.
- class list

  A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.
- class locale

  Container class for localization functionality.
  The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.
- class lock_guard

  Scoped lock idiom.
- class logic_error

  One of two subclasses of exception.
- struct logical_and

  One of the Boolean operations functors.
- struct logical_not

  One of the Boolean operations functors.
- struct logical_or

  One of the Boolean operations functors.
- class lognormal_distribution

  A lognormal_distribution random number distribution.
- struct make_signed

  make_signed
- struct make_unsigned

  make_unsigned
- class map

  A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- class mask_array

  Reference to selected subset of an array.
- class match_results

  The results of a match or search operation.
- class mem_fun1_ref_t

  One of the adaptors for member pointers.
- class mem_fun1_t

  One of the adaptors for member pointers.
- class mem_fun_ref_t

  One of the adaptors for member pointers.
- class mem_fun_t

  One of the adaptors for member pointers.
- class mersenne_twister_engine
- class messages

  Primary class template messages.
  This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.
- struct messages_base

  Messages facet base class providing catalog typedef.
- class messages_byname

class messages_byname [22.2.7.2].

- struct minus

  One of the math functors.
- struct modulus

  One of the math functors.
- class money_base

  Money format ordering data.
  This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.
- class money_get

  Primary class template money_get.
  This facet encapsulates the code to parse and return a monetary amount from a string.
- class money_put

  Primary class template money_put.
  This facet encapsulates the code to format and output a monetary amount.
- class moneypunct

  Primary class template moneypunct.
  This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.
- class moneypunct_byname

  class moneypunct_byname [22.2.6.4].
- class move_iterator
- class multimap

  A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.
- struct multiplies

  One of the math functors.
- class multiset

  A standard container made up of elements, which can be retrieved in logarithmic time.
- class mutex

  mutex
- struct negate

  One of the math functors.
- class negative_binomial_distribution

  A negative_binomial_distribution random number distribution.
- class nested_exception

  Exception class with exception_ptr data member.
- class normal_distribution

  A normal continuous distribution for random numbers.
- struct not_equal_to

  One of the comparison functors.
- class num_get

  Primary class template num_get.
  This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
- class num_put

  Primary class template num_put.
  This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.
- struct numeric_limits

  Properties of fundamental types.

- struct numeric_limits< bool >

  numeric_limits<bool> specialization.
- struct numeric_limits< char >

  numeric_limits<char> specialization.
- struct numeric_limits< char16_t >

  numeric_limits<char16_t> specialization.
- struct numeric_limits< char32_t >

  numeric_limits<char32_t> specialization.
- struct numeric_limits< double >

  numeric_limits<double> specialization.
- struct numeric_limits< float >

  numeric_limits<float> specialization.
- struct numeric_limits< int >

  numeric_limits<int> specialization.
- struct numeric_limits< long >

  numeric_limits<long> specialization.
- struct numeric_limits< long double >

  numeric_limits<long double> specialization.
- struct numeric_limits< long long >

  numeric_limits<long long> specialization.
- struct numeric_limits< short >

  numeric_limits<short> specialization.
- struct numeric_limits< signed char >

  numeric_limits<signed char> specialization.
- struct numeric_limits< unsigned char >

  numeric_limits<unsigned char> specialization.
- struct numeric_limits< unsigned int >

  numeric_limits<unsigned int> specialization.
- struct numeric_limits< unsigned long >

  numeric_limits<unsigned long> specialization.
- struct numeric_limits< unsigned long long >

  numeric_limits<unsigned long long> specialization.
- struct numeric_limits< unsigned short >

  numeric_limits<unsigned short> specialization.
- struct numeric_limits< wchar_t >

  numeric_limits<wchar_t> specialization.
- class numpunct

  Primary class template numpunct.
  This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
- class numpunct_byname

  class numpunct_byname [22.2.3.2].
- struct once_flag

  once_flag
- class ostream_iterator

  Provides output iterator semantics for streams.
- class ostreambuf_iterator

Provides output iterator semantics for streambufs.

- class out_of_range
- struct output_iterator_tag

  Marking output iterators.
- class overflow_error
- struct owner_less< shared_ptr< _Tp > >

  Partial specialization of owner_less for shared_ptr.
- struct owner_less< weak_ptr< _Tp > >

  Partial specialization of owner_less for weak_ptr.
- struct pair

  Struct holding two objects of arbitrary type.
- class piecewise_constant_distribution

  A piecewise_constant_distribution random number distribution.
- struct piecewise_construct_t

  piecewise_construct_t
- class piecewise_linear_distribution

  A piecewise_linear_distribution random number distribution.
- struct plus

  One of the math functors.
- class pointer_to_binary_function

  One of the adaptors for function pointers.
- class pointer_to_unary_function

  One of the adaptors for function pointers.
- struct pointer_traits

  Uniform interface to all pointer-like types.
- struct pointer_traits< _Tp * >

  Partial specialization for built-in pointers.
- class poisson_distribution

  A discrete Poisson random number distribution.
- class priority_queue

  A standard container automatically sorting its contents.
- class queue

  A standard container giving FIFO behavior.
- struct random_access_iterator_tag

  Random-access iterators support a superset of bidirectional iterator operations.
- class random_device
- class range_error
- struct rank

  rank
- struct ratio

  Provides compile-time rational arithmetic.
- struct ratio_equal

  ratio_equal
- struct ratio_not_equal

  ratio_not_equal
- class raw_storage_iterator
- class recursive_mutex

recursive_mutex

- class reference_wrapper

  Primary class template for reference_wrapper.
- class regex_error

  A regular expression exception class.
  The regular expression library throws objects of this class on error.
- class regex_iterator
- class regex_token_iterator
- struct regex_traits

  Class regex_traits. Describes aspects of a regular expression.
- struct remove_all_extents

  remove_all_extents
- struct remove_const

  remove_const
- struct remove_cv

  remove_cv
- struct remove_extent

  remove_extent
- struct remove_pointer

  remove_pointer
- struct remove_reference

  remove_reference
- struct remove_volatile

  remove_volatile
- class reverse_iterator
- class runtime_error

  One of two subclasses of exception.
- class scoped_allocator_adaptor

  Primary class template.
- class seed_seq

  The seed_seq class generates sequences of seeds for random number generators.
- class set

  A standard container made up of unique keys, which can be retrieved in logarithmic time.
- class shared_ptr

  A smart pointer with reference-counted copy semantics.
- class shuffle_order_engine

  Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`*.*
- class slice

  Class defining one-dimensional subset of an array.
- class slice_array

  Reference to one-dimensional subset of an array.
- class stack

  A standard container giving FILO behavior.
- class student_t_distribution

  A student_t_distribution random number distribution.
- class sub_match
- class system_error

Thrown to indicate error code of underlying system.

- class thread

  thread

- class time_base

  Time format ordering data.
  This class provides an enum representing different orderings of time: day, month, and year.

- class time_get

  Primary class template time_get.
  This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class time_get_byname

  class time_get_byname [22.2.5.2].

- class time_put

  Primary class template time_put.
  This facet encapsulates the code to format and output dates and times according to formats used by strftime().

- class time_put_byname

  class time_put_byname [22.2.5.4].

- struct try_to_lock_t

  Try to acquire ownership of the mutex without blocking.

- class tuple

  Primary class template, tuple.

- class tuple< _T1, _T2 >

  Partial specialization, 2-element tuple. Includes construction and assignment from a pair.

- struct tuple_element< 0, tuple< _Head, _Tail...> >
- struct tuple_element< __i, tuple< _Head, _Tail...> >
- struct tuple_size< tuple< _Elements...> >

  class tuple_size

- struct type_index

  Class type_index
  The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

- class type_info

  Part of RTTI.

- struct unary_function
- class unary_negate

  One of the negation functors.

- class underflow_error
- struct underlying_type

  The underlying type of an enum.

- class uniform_int_distribution

  Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ *with equal probability throughout the range.*

- class uniform_real_distribution

  Uniform continuous distribution for random numbers.

- class unique_lock

  unique_lock

- class unique_ptr

  20.7.1.2 unique_ptr for single objects.

- class unique_ptr< _Tp[ ], _Dp >

20.7.1.3 unique_ptr for array objects with a runtime length

- class unordered_map

  A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class unordered_multimap

  A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.
- class unordered_multiset

  A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class unordered_set

  A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.
- struct uses_allocator

  [allocator.uses.trait]
- struct uses_allocator< tuple< _Types...>, _Alloc >

  Partial specialization for tuples.
- class valarray

  Smart array designed to support numeric processing.
- class vector

  A standard container which offers fixed time access to individual elements in any order.
- class vector< bool, _Alloc >

  A specialization of vector for booleans which offers fixed time access to individual elements in any order.
- class weak_ptr

  A smart pointer with weak semantics.
- class weibull_distribution

  A weibull_distribution random number distribution.

Typedefs

- template<typename _Tp >
  using __allocator_base = __gnu_cxx::new_allocator< _Tp >
- typedef unsigned char **__atomic_flag_data_type**
- typedef FILE **__c_file**
- typedef __locale_t **__c_locale**
- typedef __gthread_mutex_t **__c_lock**
- template<typename _Tp , typename _Hash >
  using **__cache_default** = __not_< __and_< __is_fast_hash< _Hash >, is_default_constructible< _Hash >, is_copy_assignable< _Hash >, __detail::__is_noexcept_hash< _Tp, _Hash >>>
- template<typename _Alloc >
  using **__check_copy_constructible** = __allow_copy_cons< __is_copy_insertable< _Alloc >::value >
- template<typename _Tp >
  using **__empty_not_final** = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp >>::type
- template<typename _Tp , typename _Tp2 = typename decay<_Tp>::type>
  using **__is_socketlike** = __or_< is_integral< _Tp2 >, is_enum< _Tp2 >>
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  using **__sub_match_string** = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >

- template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
  using **__umap_hashtable** = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using __umap_traits = __detail::_Hashtable_traits< _Cache, false, true >
- template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
  using **__ummap_hashtable** = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using __ummap_traits = __detail::_Hashtable_traits< _Cache, false, false >
- template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
  using **__umset_hashtable** = _Hashtable< _Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using __umset_traits = __detail::_Hashtable_traits< _Cache, true, false >
- template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
  using **__uset_hashtable** = _Hashtable< _Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using __uset_traits = __detail::_Hashtable_traits< _Cache, true, true >
- typedef unsigned long **_Bit_type**
- template<typename _Tp1 , typename _Tp2 >
  using **_NotSame** = __not_< is_same< typename std::decay< _Tp1 >::type, typename std::decay< _Tp2 >::type >>
- template<typename... _Cond>
  using **_Require** = typename enable_if< __and_< _Cond...>::value >::type
- template<typename _InIter >
  using **_RequireInputIter** = typename enable_if< is_convertible< typename iterator_traits< _InIter >::iterator_category, input_iterator_tag >::value >::type
- typedef __atomic_base< char > atomic_char
- typedef __atomic_base< char16_t > atomic_char16_t
- typedef __atomic_base< char32_t > atomic_char32_t
- typedef __atomic_base< int > atomic_int
- typedef __atomic_base
  < int_fast16_t > atomic_int_fast16_t
- typedef __atomic_base
  < int_fast32_t > atomic_int_fast32_t
- typedef __atomic_base
  < int_fast64_t > atomic_int_fast64_t
- typedef __atomic_base
  < int_fast8_t > atomic_int_fast8_t
- typedef __atomic_base
  < int_least16_t > atomic_int_least16_t
- typedef __atomic_base
  < int_least32_t > atomic_int_least32_t
- typedef __atomic_base
  < int_least64_t > atomic_int_least64_t

- typedef __atomic_base
  < int_least8_t > atomic_int_least8_t
- typedef __atomic_base< intmax_t > atomic_intmax_t
- typedef __atomic_base< intptr_t > atomic_intptr_t
- typedef __atomic_base< long long > atomic_llong
- typedef __atomic_base< long > atomic_long
- typedef __atomic_base< ptrdiff_t > atomic_ptrdiff_t
- typedef __atomic_base< signed
  char > atomic_schar
- typedef __atomic_base< short > atomic_short
- typedef __atomic_base< size_t > atomic_size_t
- typedef __atomic_base
  < unsigned char > atomic_uchar
- typedef __atomic_base
  < unsigned int > atomic_uint
- typedef __atomic_base
  < uint_fast16_t > atomic_uint_fast16_t
- typedef __atomic_base
  < uint_fast32_t > atomic_uint_fast32_t
- typedef __atomic_base
  < uint_fast64_t > atomic_uint_fast64_t
- typedef __atomic_base
  < uint_fast8_t > atomic_uint_fast8_t
- typedef __atomic_base
  < uint_least16_t > atomic_uint_least16_t
- typedef __atomic_base
  < uint_least32_t > atomic_uint_least32_t
- typedef __atomic_base
  < uint_least64_t > atomic_uint_least64_t
- typedef __atomic_base
  < uint_least8_t > atomic_uint_least8_t
- typedef __atomic_base< uintmax_t > atomic_uintmax_t
- typedef __atomic_base< uintptr_t > atomic_uintptr_t
- typedef __atomic_base
  < unsigned long long > atomic_ullong
- typedef __atomic_base
  < unsigned long > atomic_ulong
- typedef __atomic_base
  < unsigned short > atomic_ushort
- typedef __atomic_base< wchar_t > atomic_wchar_t
- typedef match_results< const
  char ∗ > **cmatch**
- typedef regex_iterator< const
  char ∗ > **cregex_iterator**
- typedef regex_token_iterator
  < const char ∗ > cregex_token_iterator
- typedef sub_match< const char ∗ > csub_match
- typedef minstd_rand0 **default_random_engine**
- typedef integral_constant
  < bool, false > false_type
- typedef basic_filebuf< char > filebuf
- typedef basic_fstream< char > fstream

- typedef basic_ifstream< char > ifstream
- typedef basic_ios< char > ios
- typedef basic_iostream< char > iostream
- typedef basic_istream< char > istream
- typedef basic_istringstream< char > istringstream
- typedef shuffle_order_engine < minstd_rand0, 256 > **knuth_b**
- typedef enum std::memory_order memory_order
- typedef linear_congruential_engine < uint_fast32_t, 48271UL, 0UL, 2147483647UL > minstd_rand
- typedef linear_congruential_engine < uint_fast32_t, 16807UL, 0UL, 2147483647UL > minstd_rand0
- typedef mersenne_twister_engine < uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > mt19937
- typedef mersenne_twister_engine < uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > mt19937_64
- typedef void(∗ new_handler)()
- typedef basic_ofstream< char > ofstream
- typedef basic_ostream< char > ostream
- typedef basic_ostringstream< char > ostringstream
- typedef __PTRDIFF_TYPE__ **ptrdiff_t**
- typedef discard_block_engine < ranlux24_base, 223, 23 > **ranlux24**
- typedef subtract_with_carry_engine < uint_fast32_t, 24, 10, 24 > **ranlux24_base**
- typedef discard_block_engine < ranlux48_base, 389, 11 > **ranlux48**
- typedef subtract_with_carry_engine < uint_fast64_t, 48, 5, 12 > **ranlux48_base**
- template<typename _R1 , typename _R2 >
  using ratio_divide = typename __ratio_divide< _R1, _R2 >::type
- template<typename _R1 , typename _R2 >
  using ratio_multiply = typename __ratio_multiply< _R1, _R2 >::type
- typedef basic_regex< char > regex
- typedef __SIZE_TYPE__ **size_t**
- typedef match_results < string::const_iterator > **smatch**
- typedef regex_iterator < string::const_iterator > **sregex_iterator**
- typedef regex_token_iterator < string::const_iterator > sregex_token_iterator
- typedef sub_match < string::const_iterator > ssub_match
- typedef basic_streambuf< char > streambuf

- typedef long long streamoff
- typedef fpos< mbstate_t > streampos
- typedef ptrdiff_t streamsize
- typedef basic_string< char > string
- typedef basic_stringbuf< char > stringbuf
- typedef basic_stringstream< char > stringstream
- typedef void(∗ terminate_handler)()
- typedef integral_constant < bool, true > true_type
- typedef fpos< mbstate_t > u16streampos
- typedef basic_string< char16_t > u16string
- typedef fpos< mbstate_t > u32streampos
- typedef basic_string< char32_t > u32string
- typedef void(∗ unexpected_handler)()
- typedef match_results< const wchar_t ∗ > **wcmatch**
- typedef regex_iterator< const wchar_t ∗ > **wcregex_iterator**
- typedef regex_token_iterator < const wchar_t ∗ > wcregex_token_iterator
- typedef sub_match< const wchar_t ∗ > wcsub_match
- typedef basic_filebuf< wchar_t > wfilebuf
- typedef basic_fstream< wchar_t > wfstream
- typedef basic_ifstream< wchar_t > wifstream
- typedef basic_ios< wchar_t > wios
- typedef basic_iostream< wchar_t > wiostream
- typedef basic_istream< wchar_t > wistream
- typedef basic_istringstream < wchar_t > wistringstream
- typedef basic_ofstream< wchar_t > wofstream
- typedef basic_ostream< wchar_t > wostream
- typedef basic_ostringstream < wchar_t > wostringstream
- typedef basic_regex< wchar_t > wregex
- typedef match_results < wstring::const_iterator > **wsmatch**
- typedef regex_iterator < wstring::const_iterator > **wsregex_iterator**
- typedef regex_token_iterator < wstring::const_iterator > wsregex_token_iterator
- typedef sub_match < wstring::const_iterator > wssub_match
- typedef basic_streambuf< wchar_t > wstreambuf
- typedef fpos< mbstate_t > wstreampos
- typedef basic_string< wchar_t > wstring
- typedef basic_stringbuf< wchar_t > wstringbuf
- typedef basic_stringstream < wchar_t > wstringstream

Enumerations

- enum { **_S_threshold** }
- enum { **_S_chunk_size** }
- enum { **_S_word_bit** }
- enum **__memory_order_modifier** { **__memory_order_mask**, **__memory_order_modifier_mask**, **__memory_order_hle_acquire**, **__memory_order_hle_release** }
- enum **_Ios_Fmtflags** {
  _S_boolalpha, **_S_dec**, **_S_fixed**, **_S_hex**,
  _S_internal, **_S_left**, **_S_oct**, **_S_right**,
  _S_scientific, **_S_showbase**, **_S_showpoint**, **_S_showpos**,
  _S_skipws, **_S_unitbuf**, **_S_uppercase**, **_S_adjustfield**,
  _S_basefield, **_S_floatfield**, **_S_ios_fmtflags_end** }
- enum **_Ios_Iostate** {
  _S_goodbit, **_S_badbit**, **_S_eofbit**, **_S_failbit**,
  _S_ios_iostate_end }
- enum **_Ios_Openmode** {
  _S_app, **_S_ate**, **_S_bin**, **_S_in**,
  _S_out, **_S_trunc**, **_S_ios_openmode_end** }
- enum **_Ios_Seekdir** { **_S_beg**, **_S_cur**, **_S_end**, **_S_ios_seekdir_end** }
- enum **_Manager_operation** { **__get_type_info**, **__get_functor_ptr**, **__clone_functor**, **__destroy_functor** }
- enum **_Rb_tree_color** { **_S_red**, **_S_black** }
- enum cv_status
- enum **errc**
- enum float_denorm_style { denorm_indeterminate, denorm_absent, denorm_present }
- enum float_round_style {
  round_indeterminate, round_toward_zero, round_to_nearest, round_toward_infinity,
  round_toward_neg_infinity }
- enum future_errc
- enum future_status
- enum launch
- enum memory_order {
  memory_order_relaxed, **memory_order_consume**, **memory_order_acquire**, **memory_order_release**,
  memory_order_acq_rel, **memory_order_seq_cst** }

Functions

- template<typename _CharT >
  _CharT ∗ **__add_grouping** (_CharT ∗__s, _CharT __sep, const char ∗__gbeg, size_t __gsize, const _CharT ∗__first, const _CharT ∗__last)
- template<typename _Tp >
  _Tp ∗ __addressof (_Tp &__r) noexcept
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp >
  void **__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp , typename _Compare >
  void **__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)
- template<typename _InputIterator , typename _Distance >
  void **__advance** (_InputIterator &__i, _Distance __n, input_iterator_tag)
- template<typename _BidirectionalIterator , typename _Distance >
  void **__advance** (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)

- template<typename _RandomAccessIterator , typename _Distance >
  void **__advance** (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)
- template<typename _Alloc >
  void **__alloc_on_copy** (_Alloc &__one, const _Alloc &__two)
- template<typename _Alloc >
  _Alloc **__alloc_on_copy** (const _Alloc &__a)
- template<typename _Alloc >
  void **__alloc_on_move** (_Alloc &__one, _Alloc &__two)
- template<typename _Alloc >
  void **__alloc_on_swap** (_Alloc &__one, _Alloc &__two)
- template<typename _Tp , _Lock_policy _Lp, typename _Alloc , typename... _Args>
  __shared_ptr< _Tp, _Lp > **__allocate_shared** (const _Alloc &__a, _Args &&...__args)
- template<typename _Callable , typename... _Args>
  _Bind_simple_helper< _Callable, _Args...>::__type **__bind_simple** (_Callable &&__callable, _Args &&...__args)
- template<typename _Functor >
  _Functor & **__callable_functor** (_Functor &__f)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::* > **__callable_functor** (_Member _Class::*&__p)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::* > **__callable_functor** (_Member _Class::*const &__p)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::* > **__callable_functor** (_Member _Class::*volatile &__p)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::* > **__callable_functor** (_Member _Class::*const volatile &__p)
- template<typename _Facet >
  const _Facet & **__check_facet** (const _Facet *__f)
- template<typename _RandomAccessIterator , typename _Distance >
  void **__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size)
- template<typename _RandomAccessIterator , typename _Distance , typename _Compare >
  void **__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)
- constexpr memory_order **__cmpexch_failure_order** (memory_order __m) noexcept
- constexpr memory_order **__cmpexch_failure_order2** (memory_order __m) noexcept
- template<typename _Tp >
  _Tp **__complex_abs** (const complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_acos** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  _Tp **__complex_arg** (const complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_asin** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_asinh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_atan** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_cos** (const complex< _Tp > &__z)

- template<typename _Tp >
  complex< _Tp > **__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  std::complex< _Tp > **__complex_proj** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **__complex_tanh** (const complex< _Tp > &__z)
- int **__convert_from_v** (const __c_locale &__cloc __attribute__((__unused__)), char ∗__out, const int __size __attribute__((__unused__)), const char ∗__fmt,...)
- template<typename _Tp >
  void **__convert_to_v** (const char ∗, _Tp &, ios_base::iostate &, const __c_locale &) throw ()
- template<>
  void **__convert_to_v** (const char ∗, float &, ios_base::iostate &, const __c_locale &) throw ()
- template<>
  void **__convert_to_v** (const char ∗, double &, ios_base::iostate &, const __c_locale &) throw ()
- template<>
  void **__convert_to_v** (const char ∗, long double &, ios_base::iostate &, const __c_locale &) throw ()
- template<bool _IsMove, typename _II , typename _OI >
  _OI **__copy_move_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **__copy_move_a2** (_CharT ∗__first, _CharT ∗__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **__copy_move_a2** (const _CharT ∗__first, const _CharT ∗__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  _CharT ∗ >::__type **__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT ∗__result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT,
  char_traits< _CharT >
  > >::__type **__copy_move_a2** (_CharT ∗, _CharT ∗, ostreambuf_iterator< _CharT, char_traits< _CharT > >)

- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT,
  char_traits< _CharT >
  > >::__type **__copy_move_a2** (const _CharT ∗, const _CharT ∗, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  _CharT ∗ >::__type **__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT ∗)
- template<bool _IsMove, typename _II , typename _OI >
  _OI **__copy_move_a2** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _BI1 , typename _BI2 >
  _BI2 **__copy_move_backward_a** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _BI1 , typename _BI2 >
  _BI2 **__copy_move_backward_a2** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  _OutputIterator **__copy_n** (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Size , typename _OutputIterator >
  _OutputIterator **__copy_n** (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)
- template<typename _CharT , typename _Traits >
  streamsize **__copy_streambufs** (basic_streambuf< _CharT, _Traits > ∗__sbin, basic_streambuf< _CharT, _Traits > ∗__sbout)
- template<typename _CharT , typename _Traits >
  streamsize **__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > ∗, basic_streambuf< _CharT, _Traits > ∗, bool &)
- template<>
  streamsize **__copy_streambufs_eof** (basic_streambuf< char > ∗__sbin, basic_streambuf< char > ∗__sbout, bool &__ineof)
- template<>
  streamsize **__copy_streambufs_eof** (basic_streambuf< wchar_t > ∗__sbin, basic_streambuf< wchar_t > ∗__sbout, bool &__ineof)
- size_t **__deque_buf_size** (size_t __size)
- template<typename _InputIterator >
  iterator_traits
  < _InputIterator >
  ::difference_type **__distance** (_InputIterator __first, _InputIterator __last, input_iterator_tag)
- template<typename _RandomAccessIterator >
  iterator_traits
  < _RandomAccessIterator >
  ::difference_type **__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _Alloc >
  void **__do_alloc_on_copy** (_Alloc &__one, const _Alloc &__two, true_type)
- template<typename _Alloc >
  void **__do_alloc_on_move** (_Alloc &__one, _Alloc &__two, true_type)
- template<typename _Alloc >
  void **__do_alloc_on_swap** (_Alloc &__one, _Alloc &__two, true_type)
- template<typename _Alloc >
  auto **__do_outermost** (_Alloc &__a, _Alloc ∗) -> decltype(__a.outer_allocator())

- template<typename _Alloc >
  _Alloc & **__do_outermost** (_Alloc &__a,...)
- template<_Lock_policy _Lp, typename _Tp1 , typename _Tp2 >
  void **__enable_shared_from_this_helper** (const __shared_count< _Lp > &, const __enable_shared_from_this< _Tp1, _Lp > ∗, const _Tp2 ∗) noexcept
- template<typename _Tp1 , typename _Tp2 >
  void **__enable_shared_from_this_helper** (const __shared_count<> &, const enable_shared_from_this< _Tp1 > ∗, const _Tp2 ∗) noexcept
- template<_Lock_policy _Lp>
  void **__enable_shared_from_this_helper** (const __shared_count< _Lp > &,...) noexcept
- template<typename _II1 , typename _II2 >
  bool **__equal_aux** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _ForwardIterator , typename _Tp >
  __gnu_cxx::__enable_if
  <!__is_scalar< _Tp >::__value,
  void >::__type **__fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _Tp >
  __gnu_cxx::__enable_if
  < __is_byte< _Tp >::__value,
  void >::__type **__fill_a** (_Tp ∗__first, _Tp ∗__last, const _Tp &__c)
- void **__fill_bvector** (_Bit_iterator __first, _Bit_iterator __last, bool __x)
- template<typename _OutputIterator , typename _Size , typename _Tp >
  __gnu_cxx::__enable_if
  <!__is_scalar< _Tp >::__value,
  _OutputIterator >::__type **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value)
- template<typename _Size , typename _Tp >
  __gnu_cxx::__enable_if
  < __is_byte< _Tp >::__value,
  _Tp ∗ >::__type **__fill_n_a** (_Tp ∗__first, _Size __n, const _Tp &__c)
- template<typename _RandomAccessIterator >
  void __final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void __final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _Tp >
  _InputIterator __find (_InputIterator __first, _InputIterator __last, const _Tp &__val, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Tp >
  _RandomAccessIterator __find (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 **__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 **__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 >
  _BidirectionalIterator1 **__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BinaryPredicate >
  _BidirectionalIterator1 **__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator __find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)

- template<typename _RandomAccessIterator , typename _Predicate >
  _RandomAccessIterator __find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Predicate >
  _RandomAccessIterator __find_if_not (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Predicate , typename _Distance >
  _InputIterator __find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)
- template<typename _EuclideanRingElement >
  _EuclideanRingElement __gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)
- template<std::size_t __i, typename _Head , typename... _Tail>
  constexpr __add_ref< _Head >::type **__get_helper** (_Tuple_impl< __i, _Head, _Tail...> &__t) noexcept
- template<std::size_t __i, typename _Head , typename... _Tail>
  constexpr __add_c_ref< _Head >
  ::type **__get_helper** (const _Tuple_impl< __i, _Head, _Tail...> &__t) noexcept
- template<typename _Ex >
  const nested_exception ∗ **__get_nested_exception** (const _Ex &__ex)
- mutex & **__get_once_mutex** ()
- template<typename _RandomAccessIterator >
  void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void __heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Tp >
  size_t **__iconv_adaptor** (size_t(∗__func)(iconv_t, _Tp, size_t ∗, char ∗∗, size_t ∗), iconv_t __cd, char ∗∗__inbuf, size_t ∗__inbytes, char ∗∗__outbuf, size_t ∗__outbytes)
- template<typename _ForwardIterator , typename _Predicate , typename _Distance >
  _ForwardIterator __inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)
- template<typename _RandomAccessIterator >
  void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void __insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _CharT , typename _ValueT >
  int **__int_to_char** (_CharT ∗__bufend, _ValueT __v, const _CharT ∗__lit, ios_base::fmtflags __flags, bool __dec)
- template<typename _RandomAccessIterator , typename _Size >
  void **__introselect** (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)
- template<typename _RandomAccessIterator , typename _Size , typename _Compare >
  void **__introselect** (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Size >
  void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)

- template<typename _RandomAccessIterator , typename _Size , typename _Compare >
  void __introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)
- template<typename _Functor , typename... _Args>
  enable_if< (!is_member_pointer
  < _Functor >::value
  &&!is_function< _Functor >
  ::value &&!is_function
  < typename remove_pointer
  < _Functor >::type >::value),
  typename result_of< _Functor(_Args &&...)>
  ::type >::type __invoke (_Functor &__f, _Args &&...__args)
- template<typename _Functor , typename... _Args>
  enable_if< (is_pointer
  < _Functor >::value
  &&is_function< typename
  remove_pointer< _Functor >
  ::type >::value), typename
  result_of< _Functor(_Args &&...)>
  ::type >::type **__invoke** (_Functor __f, _Args &&...__args)
- template<typename _RandomAccessIterator , typename _Distance >
  bool **__is_heap** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator , typename _Compare , typename _Distance >
  bool **__is_heap** (_RandomAccessIterator __first, _Compare __comp, _Distance __n)
- template<typename _RandomAccessIterator >
  bool **__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  bool **__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Distance >
  _Distance **__is_heap_until** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator , typename _Distance , typename _Compare >
  _Distance **__is_heap_until** (_RandomAccessIterator __first, _Distance __n, _Compare __comp)
- template<typename _Iter >
  iterator_traits< _Iter >
  ::iterator_category __iterator_category (const _Iter &)
- template<typename _II1 , typename _II2 >
  bool **__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- constexpr int __lg (int __n)
- constexpr unsigned **__lg** (unsigned __n)
- constexpr long **__lg** (long __n)
- constexpr unsigned long **__lg** (unsigned long __n)
- constexpr long long **__lg** (long long __n)
- constexpr unsigned long long **__lg** (unsigned long long __n)
- template<typename _Iterator , typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>
  _ReturnType **__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Tp , _Lock_policy _Lp, typename... _Args>
  __shared_ptr< _Tp, _Lp > **__make_shared** (_Args &&...__args)
- template<typename _BidirectionalIterator , typename _Distance , typename _Pointer >
  void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)

- template<typename _BidirectionalIterator , typename _Distance , typename _Pointer , typename _Compare >
  void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance >
  void **__merge_sort_loop** (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size)
- template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance , typename _Compare >
  void **__merge_sort_loop** (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Pointer >
  void **__merge_sort_with_buffer** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer)
- template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >
  void **__merge_sort_with_buffer** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)
- template<typename _BidirectionalIterator , typename _Distance >
  void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2)
- template<typename _BidirectionalIterator , typename _Distance , typename _Compare >
  void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)
- template<typename _Iterator >
  _Miter_base< _Iterator >
  ::iterator_type **__miter_base** (_Iterator __it)
- template<typename _Iterator >
  void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)
- template<typename _Iterator , typename _Compare >
  void __move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator __move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator __move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 >
  void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 , typename _Compare >
  void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)
- template<typename _Iterator >
  _Niter_base< _Iterator >
  ::iterator_type **__niter_base** (_Iterator __it)
- void **__once_proxy** (void)
- template<typename _CharT , typename _Traits >
  void **__ostream_fill** (basic_ostream< _CharT, _Traits > &__out, streamsize __n)

- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **__ostream_insert** (basic_ostream< _CharT, _Traits > &__out, const _CharT ∗__s, streamsize __n)
- template<typename _CharT , typename _Traits >
  void **__ostream_write** (basic_ostream< _CharT, _Traits > &__out, const _CharT ∗__s, streamsize __n)
- template<typename _Alloc >
  auto **__outermost** (_Alloc &__a) -> decltype(__do_outermost(__a,&__a))
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)
- template<typename _BidirectionalIterator , typename _Predicate >
  _BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator >
  void **__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result)
- template<typename _RandomAccessIterator , typename _Compare >
  void **__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp >
  void **__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp , typename _Compare >
  void **__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)
- template<typename _BidirectionalIterator >
  void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator >
  void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _ForwardIterator >
  void __rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)
- template<typename _BidirectionalIterator >
  void __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator >
  void __rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance >
  _BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)
- template<typename _ForwardIterator , typename _Integer , typename _Tp >
  _ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, std::forward_iterator_tag)
- template<typename _RandomAccessIter , typename _Integer , typename _Tp >
  _RandomAccessIter __search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_access_iterator_tag)
- template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)
- template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _RandomAccessIter __search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)
- void **__set_once_functor_lock_ptr** (unique_lock< mutex > ∗)

- template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance >
  _ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)
- template<typename _RandomAccessIterator , typename _Pointer , typename _Distance >
  void **__stable_sort_adaptive** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)
- template<typename _RandomAccessIterator , typename _Pointer , typename _Distance , typename _Compare >
  void **__stable_sort_adaptive** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- void **__throw_bad_alloc** (void) __attribute__((__noreturn__))
- void **__throw_bad_cast** (void) __attribute__((__noreturn__))
- void **__throw_bad_exception** (void) __attribute__((__noreturn__))
- void **__throw_bad_function_call** () __attribute__((__noreturn__))
- void **__throw_bad_typeid** (void) __attribute__((__noreturn__))
- void **__throw_bad_weak_ptr** ()
- void **__throw_domain_error** (const char ∗) __attribute__((__noreturn__))
- void **__throw_future_error** (int) __attribute__((__noreturn__))
- void **__throw_invalid_argument** (const char ∗) __attribute__((__noreturn__))
- void **__throw_ios_failure** (const char ∗) __attribute__((__noreturn__))
- void **__throw_length_error** (const char ∗) __attribute__((__noreturn__))
- void **__throw_logic_error** (const char ∗) __attribute__((__noreturn__))
- void **__throw_out_of_range** (const char ∗) __attribute__((__noreturn__))
- void **__throw_overflow_error** (const char ∗) __attribute__((__noreturn__))
- void **__throw_range_error** (const char ∗) __attribute__((__noreturn__))
- void **__throw_regex_error** (regex_constants::error_type __ecode)
- void **__throw_runtime_error** (const char ∗) __attribute__((__noreturn__))
- void **__throw_system_error** (int) __attribute__((__noreturn__))
- void **__throw_underflow_error** (const char ∗) __attribute__((__noreturn__))
- template<typename _Ex >
  void **__throw_with_nested** (_Ex &&, const nested_exception ∗=0) __attribute__((__noreturn__))
- template<typename _Ex >
  void **__throw_with_nested** (_Ex &&,...) __attribute__((__noreturn__))
- template<typename _Lock >
  unique_lock< _Lock > **__try_to_lock** (_Lock &__l)
- template<typename _RandomAccessIterator >
  void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void __unguarded_linear_insert (_RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void __unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Tp >
  _RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)
- template<typename _RandomAccessIterator , typename _Tp , typename _Compare >
  _RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)
- template<typename _RandomAccessIterator >
  _RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)

- template<typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Pointer , typename _ForwardIterator >
  void **__uninitialized_construct_buf** (_Pointer __first, _Pointer __last, _ForwardIterator __seed)
- template<typename _InputIterator , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **__uninitialized_copy_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
- template<typename _InputIterator , typename _ForwardIterator , typename _Tp >
  _ForwardIterator **__uninitialized_copy_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **__uninitialized_copy_move** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)
- template<typename _InputIterator , typename _Size , typename _ForwardIterator >
  _ForwardIterator **__uninitialized_copy_n** (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Size , typename _ForwardIterator >
  _ForwardIterator **__uninitialized_copy_n** (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)
- template<typename _ForwardIterator >
  void **__uninitialized_default** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Allocator >
  void **__uninitialized_default_a** (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Tp >
  void **__uninitialized_default_a** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)
- template<typename _ForwardIterator , typename _Size >
  void **__uninitialized_default_n** (_ForwardIterator __first, _Size __n)
- template<typename _ForwardIterator , typename _Size , typename _Allocator >
  void **__uninitialized_default_n_a** (_ForwardIterator __first, _Size __n, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Size , typename _Tp >
  void **__uninitialized_default_n_a** (_ForwardIterator __first, _Size __n, allocator< _Tp > &)
- template<typename _ForwardIterator , typename _Tp , typename _Allocator >
  void **__uninitialized_fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Tp , typename _Tp2 >
  void **__uninitialized_fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator< _Tp2 > &)
- template<typename _ForwardIterator , typename _Tp , typename _InputIterator , typename _Allocator >
  _ForwardIterator **__uninitialized_fill_move** (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Size , typename _Tp , typename _Allocator >
  void **__uninitialized_fill_n_a** (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Size , typename _Tp , typename _Tp2 >
  void **__uninitialized_fill_n_a** (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator< _Tp2 > &)
- template<typename _InputIterator , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **__uninitialized_move_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **__uninitialized_move_copy** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)

- template<typename _InputIterator , typename _ForwardIterator , typename _Tp , typename _Allocator >
  void **__uninitialized_move_fill** (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)
- template<typename _InputIterator , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **__uninitialized_move_if_noexcept_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _OutputIterator >
  _OutputIterator **__unique_copy** (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator **__unique_copy** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _ForwardIterator >
  _ForwardIterator **__unique_copy** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)
- template<typename _ForwardIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator **__unique_copy** (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator **__unique_copy** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator **__unique_copy** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)
- template<typename _Bi_iter >
  const sub_match< _Bi_iter > & **__unmatched_sub** ()
- template<typename _Tp , typename _Alloc , typename... _Args>
  __uses_alloc_impl< _Tp, _Alloc,
  _Args...> **__use_alloc** (const _Alloc &__a)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)
- template<typename _Tp , class _Dom >
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp , class _Dom >
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)
- template<typename _Tp , class _Dom >
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)
- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __a, size_t __n, _Tp ∗__restrict__ __b)
- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __a, size_t __n, size_t __s, _Tp ∗__restrict__ __b)

- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __a, _Tp ∗__restrict__ __b, size_t __n, size_t __s)
- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __src, size_t __n, size_t __s1, _Tp ∗__restrict__ __dst, size_t __s2)
- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __a, const size_t ∗__restrict__ __i, _Tp ∗__restrict__ __b, size_t __n)
- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __a, size_t __n, _Tp ∗__restrict__ __b, const size_t ∗__restrict__ __i)
- template<typename _Tp >
  void **__valarray_copy** (const _Tp ∗__restrict__ __src, size_t __n, const size_t ∗__restrict__ __i, _Tp ∗__restrict__ __dst, const size_t ∗__restrict__ __j)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **__valarray_copy** (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)
- template<typename _Tp >
  void **__valarray_copy_construct** (const _Tp ∗__b, const _Tp ∗__e, _Tp ∗__restrict__ __o)
- template<typename _Tp >
  void **__valarray_copy_construct** (const _Tp ∗__restrict__ __a, size_t __n, size_t __s, _Tp ∗__restrict__ __o)
- template<typename _Tp >
  void **__valarray_copy_construct** (const _Tp ∗__restrict__ __a, const size_t ∗__restrict__ __i, _Tp ∗__restrict__ __o, size_t __n)
- template<typename _Tp , class _Dom >
  void **__valarray_copy_construct** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp >
  void **__valarray_copy_construct** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **__valarray_copy_construct** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **__valarray_copy_construct** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **__valarray_default_construct** (_Tp ∗__b, _Tp ∗__e)
- template<typename _Tp >
  void **__valarray_destroy_elements** (_Tp ∗__b, _Tp ∗__e)
- template<typename _Tp >
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)
- template<typename _Tp >
  void **__valarray_fill** (_Tp ∗__restrict__ __a, size_t __n, const _Tp &__t)

- template<typename _Tp >
  void **__valarray_fill** (_Tp ∗__restrict__ __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp >
  void **__valarray_fill** (_Tp ∗__restrict__ __a, const size_t ∗__restrict__ __i, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp >
  void **__valarray_fill** (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **__valarray_fill_construct** (_Tp ∗__b, _Tp ∗__e, const _Tp __t)
- void ∗ **__valarray_get_memory** (size_t __n)
- template<typename _Tp >
  _Tp ∗__restrict__ **__valarray_get_storage** (size_t __n)
- template<typename _Ta >
  _Ta::value_type **__valarray_max** (const _Ta &__a)
- template<typename _Ta >
  _Ta::value_type **__valarray_min** (const _Ta &__a)
- template<typename _Tp >
  _Tp **__valarray_product** (const _Tp ∗__f, const _Tp ∗__l)
- void **__valarray_release_memory** (void ∗__p)
- template<typename _Tp >
  _Tp **__valarray_sum** (const _Tp ∗__f, const _Tp ∗__l)
- bool **__verify_grouping** (const char ∗__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()
- template<std::size_t _Ind, typename... _Tp>
  auto **__volget** (volatile tuple< _Tp...> &__tuple) -> typename tuple_element< _Ind, tuple< _Tp...>>::type volatile &
- template<std::size_t _Ind, typename... _Tp>
  auto **__volget** (const volatile tuple< _Tp...> &__tuple) -> typename tuple_element< _Ind, tuple< _Tp...>>::type const volatile &
- template<typename _CharT >
  ostreambuf_iterator< _CharT > **__write** (ostreambuf_iterator< _CharT > __s, const _CharT ∗__ws, int __len)
- template<typename _CharT , typename _OutIter >
  _OutIter **__write** (_OutIter __s, const _CharT ∗__ws, int __len)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)

- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___divides** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp >
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___minus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)

- template<typename _Tp , class _Dom >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___plus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)

- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)

- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _T1 , typename... _Args>
  void _Construct (_T1 ∗__p, _Args &&...__args)
- template<typename _Tp >
  void _Destroy (_Tp ∗__pointer)
- template<typename _ForwardIterator >
  void _Destroy (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Allocator >
  void **_Destroy** (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Tp >
  void **_Destroy** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)
- size_t _Find_first () const noexcept
- size_t _Find_next (size_t __prev) const noexcept
- size_t **_Fnv_hash_bytes** (const void ∗__ptr, size_t __len, size_t __seed)
- size_t **_Hash_bytes** (const void ∗__ptr, size_t __len, size_t __seed)
- template<class _CharT , class _Traits >
  void **_M_copy_from_ptr** (const _CharT ∗, size_t, size_t, size_t, _CharT, _CharT)
- template<class _CharT , class _Traits , class _Alloc >
  void **_M_copy_from_string** (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)
- template<class _CharT , class _Traits , class _Alloc >
  void **_M_copy_from_string** (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n)
- template<class _CharT , class _Traits , class _Alloc >
  void **_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc > &, _CharT, _CharT) const
- template<class _CharT , class _Traits , class _Alloc >
  void **_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc > &__s) const
- template<size_t _Nb>
  _M_do_and (__rhs)
- unsigned int **_Rb_tree_black_count** (const _Rb_tree_node_base ∗__node, const _Rb_tree_node_base ∗__root) throw ()

- _Rb_tree_node_base ∗ **_Rb_tree_decrement** (_Rb_tree_node_base ∗__x) throw ()
- const _Rb_tree_node_base ∗ **_Rb_tree_decrement** (const _Rb_tree_node_base ∗__x) throw ()
- _Rb_tree_node_base ∗ **_Rb_tree_increment** (_Rb_tree_node_base ∗__x) throw ()
- const _Rb_tree_node_base ∗ **_Rb_tree_increment** (const _Rb_tree_node_base ∗__x) throw ()
- void **_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base ∗__x, _Rb_tree_node_base ∗__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base ∗ **_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base ∗const __z, _Rb_tree_node_base &__header) throw ()
- void **abort** (void) throw ()
- template<typename _Tp >
  _Tp abs (const complex< _Tp > &)
- constexpr double **abs** (double __x)
- constexpr float **abs** (float __x)
- constexpr long double **abs** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **abs** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Abs, _Expr,
  _Dom >, typename
  _Dom::value_type > **abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Abs,
  _ValArray, _Tp >, _Tp > **abs** (const valarray< _Tp > &__v)
- template<typename _InputIterator , typename _Tp >
  _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator , typename _Tp , typename _BinaryOperation >
  _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- constexpr float **acos** (float __x)
- constexpr long double **acos** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **acos** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Acos, _Expr,
  _Dom >, typename
  _Dom::value_type > **acos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Acos,
  _ValArray, _Tp >, _Tp > **acos** (const valarray< _Tp > &__v)
- template<typename _Tp >
  std::complex< _Tp > acos (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > acosh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  _Tp ∗ addressof (_Tp &__r) noexcept
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)

- template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _FIter >
  _FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter , typename _BinaryPredicate >
  _FIter **adjacent_find** (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator >
  _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator , typename _Distance >
  void advance (_InputIterator &__i, _Distance __n)
- bool all () const noexcept
- template<typename _IIter , typename _Predicate >
  bool **all_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
  bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp , typename _Alloc , typename... _Args>
  shared_ptr< _Tp > allocate_shared (const _Alloc &__a, _Args &&...__args)
- bool any () const noexcept
- template<typename _IIter , typename _Predicate >
  bool **any_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
  bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp >
  _Tp arg (const complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)
- constexpr float **asin** (float __x)
- constexpr long double **asin** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **asin** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Asin, _Expr,
  _Dom >, typename
  _Dom::value_type > **asin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Asin,
  _ValArray, _Tp >, _Tp > **asin** (const valarray< _Tp > &__v)
- template<typename _Tp >
  std::complex< _Tp > asin (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > asinh (const std::complex< _Tp > &__z)
- template<typename _Fn , typename... _Args>
  future< typename result_of
  < _Fn(_Args...)>::type > **async** (launch __policy, _Fn &&__fn, _Args &&...__args)

- template<typename _Fn , typename... _Args>
  future< typename result_of
  < _Fn(_Args...)>::type > **async** (_Fn &&__fn, _Args &&...__args)
- constexpr float **atan** (float __x)
- constexpr long double **atan** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **atan** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Atan, _Expr,
  _Dom >, typename
  _Dom::value_type > **atan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Atan,
  _ValArray, _Tp >, _Tp > **atan** (const valarray< _Tp > &__v)
- template<typename _Tp >
  std::complex< _Tp > atan (const std::complex< _Tp > &__z)
- constexpr float **atan2** (float __y, float __x)
- constexpr long double **atan2** (long double __y, long double __x)
- template<typename _Tp , typename _Up >
  constexpr
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **atan2** (_Tp __y, _Up __x)
- template<typename _Tp >
  _Expr< _BinClos< _Atan2,
  _ValArray, _ValArray, _Tp, _Tp >
  , _Tp > **atan2** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom >
  _Expr< _BinClos< _Atan2, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< _Atan2, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename _Dom1::value_type > **atan2** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom >
  _Expr< _BinClos< _Atan2, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< _Atan2,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **atan2** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<typename _Tp >
  _Expr< _BinClos< _Atan2,
  _ValArray, _Constant, _Tp, _Tp >
  , _Tp > **atan2** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< _Atan2,
  _Constant, _ValArray, _Tp, _Tp >
  , _Tp > **atan2** (const _Tp &__t, const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _BinClos< _Atan2,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **atan2** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  std::complex< _Tp > atanh (const std::complex< _Tp > &__z)
- int **atexit** (void(∗)()) throw ()
- template<typename _ITp >
  bool **atomic_compare_exchange_strong** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_strong** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_strong_explicit** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_weak** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_weak** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_weak_explicit** (atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > ∗__a, _ITp ∗__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  _ITp **atomic_exchange** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_exchange** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_exchange_explicit** (atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_exchange_explicit** (volatile atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_add** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_add** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_add** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_add** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept

- template<typename _ITp >
  _ITp **atomic_fetch_add_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_add_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_add_explicit** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_add_explicit** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_and** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_and** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_and_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_or** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_or** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_or_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_sub** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_sub** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_sub** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_sub** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_sub_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_sub_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_sub_explicit** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **atomic_fetch_sub_explicit** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_xor** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_xor** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_xor_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- void **atomic_flag_clear** (atomic_flag ∗__a) noexcept

- void **atomic_flag_clear** (volatile atomic_flag ∗__a) noexcept
- void **atomic_flag_clear_explicit** (atomic_flag ∗__a, memory_order __m) noexcept
- void **atomic_flag_clear_explicit** (volatile atomic_flag ∗__a, memory_order __m) noexcept
- bool **atomic_flag_test_and_set** (atomic_flag ∗__a) noexcept
- bool **atomic_flag_test_and_set** (volatile atomic_flag ∗__a) noexcept
- bool **atomic_flag_test_and_set_explicit** (atomic_flag ∗__a, memory_order __m) noexcept
- bool **atomic_flag_test_and_set_explicit** (volatile atomic_flag ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  void **atomic_init** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **atomic_init** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  bool **atomic_is_lock_free** (const atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  bool **atomic_is_lock_free** (const volatile atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **atomic_load** (const atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **atomic_load** (const volatile atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **atomic_load_explicit** (const atomic< _ITp > ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **atomic_load_explicit** (const volatile atomic< _ITp > ∗__a, memory_order __m) noexcept
- void **atomic_signal_fence** (memory_order __m) noexcept
- template<typename _ITp >
  void **atomic_store** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **atomic_store** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **atomic_store_explicit** (atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  void **atomic_store_explicit** (volatile atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- void **atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Container >
  back_insert_iterator< _Container > back_inserter (_Container &__x)
- template<class _Container >
  auto begin (_Container &__cont) -> decltype(__cont.begin())
- template<class _Container >
  auto begin (const _Container &__cont) -> decltype(__cont.begin())
- template<class _Tp , size_t _Nm>
  _Tp ∗ begin (_Tp(&__arr)[_Nm])
- template<class _Tp >
  constexpr const _Tp ∗ begin (initializer_list< _Tp > __ils) noexcept
- template<class _Tp >
  _Tp ∗ begin (valarray< _Tp > &__va)
- template<class _Tp >
  const _Tp ∗ begin (const valarray< _Tp > &__va)
- template<typename _FIter , typename _Tp >
  bool **binary_search** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  bool **binary_search** (_FIter, _FIter, const _Tp &, _Compare)

- template<typename _ForwardIterator , typename _Tp >
  bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Func , typename... _BoundArgs>
  _Bind_helper< __is_socketlike
  < _Func >::value, _Func,
  _BoundArgs...>::type bind (_Func &&__f, _BoundArgs &&...__args)
- template<typename _Result , typename _Func , typename... _BoundArgs>
  _Bindres_helper< _Result,
  _Func, _BoundArgs...>::type bind (_Func &&__f, _BoundArgs &&...__args)
- template<typename _Operation , typename _Tp >
  binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation , typename _Tp >
  binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)
- ios_base & boolalpha (ios_base &__base)
- template<typename _Callable , typename... _Args>
  void call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)
- constexpr float **ceil** (float __x)
- constexpr long double **ceil** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **ceil** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > conj (const complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **conj** (_Tp __x)
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _IIter , typename _OIter >
  _OIter **copy** (_IIter, _IIter, _OIter)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **copy** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _II , typename _OI >
  _OI copy (_II __first, _II __last, _OI __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **copy** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _BIter1 , typename _BIter2 >
  _BIter2 **copy_backward** (_BIter1, _BIter1, _BIter2)

- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **copy_backward** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _BI1 , typename _BI2 >
  _BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **copy_backward** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Ex >
  exception_ptr copy_exception (_Ex __ex) noexcept
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
  _OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _IIter , typename _Size , typename _OIter >
  _OIter **copy_n** (_IIter, _Size, _OIter)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  _OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _Tp >
  complex< _Tp > cos (const complex< _Tp > &)
- constexpr float **cos** (float __x)
- constexpr long double **cos** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **cos** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Cos, _Expr,
  _Dom >, typename
  _Dom::value_type > **cos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Cos,
  _ValArray, _Tp >, _Tp > **cos** (const valarray< _Tp > &__v)
- template<typename _Tp >
  complex< _Tp > cosh (const complex< _Tp > &)
- constexpr float **cosh** (float __x)
- constexpr long double **cosh** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **cosh** (_Tp __x)
- template<typename _Tp >
  _Expr< _UnClos< _Cosh,
  _ValArray, _Tp >, _Tp > **cosh** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Cosh, _Expr,
  _Dom >, typename
  _Dom::value_type > **cosh** (const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **count** (_IIter, _IIter, const _Tp &)
- size_t count () const noexcept
- template<typename _InputIterator , typename _Tp >
  iterator_traits
  < _InputIterator >
  ::difference_type count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **count_if** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
  iterator_traits
  < _InputIterator >
  ::difference_type count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- exception_ptr current_exception () noexcept
- ios_base & dec (ios_base &__base)
- template<typename _Tp >
  add_rvalue_reference< _Tp >::type declval () noexcept
- template<typename _InputIterator >
  iterator_traits
  < _InputIterator >
  ::difference_type distance (_InputIterator __first, _InputIterator __last)
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<class _Container >
  auto end (_Container &__cont) -> decltype(__cont.end())
- template<class _Container >
  auto end (const _Container &__cont) -> decltype(__cont.end())
- template<class _Tp , size_t _Nm>
  _Tp ∗ end (_Tp(&__arr)[_Nm])
- template<class _Tp >
  constexpr const _Tp ∗ end (initializer_list< _Tp > __ils) noexcept
- template<class _Tp >
  _Tp ∗ end (valarray< _Tp > &__va)
- template<class _Tp >
  const _Tp ∗ end (const valarray< _Tp > &__va)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > &__os)
- template<typename _IIter1 , typename _IIter2 >
  bool **equal** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
  bool equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _II1 , typename _II2 >
  bool equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _FIter , typename _Tp >
  pair< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  pair< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &, _Compare)

- template<typename _ForwardIterator , typename _Tp >
  pair< _ForwardIterator,
  _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  pair< _ForwardIterator,
  _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- void **exit** (int) throw ()
- template<typename _Tp >
  complex< _Tp > exp (const complex< _Tp > &)
- constexpr float **exp** (float __x)
- constexpr long double **exp** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **exp** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Exp, _Expr,
  _Dom >, typename
  _Dom::value_type > **exp** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Exp,
  _ValArray, _Tp >, _Tp > **exp** (const valarray< _Tp > &__v)
- constexpr float **fabs** (float __x)
- constexpr long double **fabs** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **fabs** (_Tp __x)
- template<typename _Tp >
  _Tp fabs (const std::complex< _Tp > &__z)
- template<typename _FIter , typename _Tp >
  void **fill** (_FIter, _FIter, const _Tp &)
- void **fill** (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)
- template<typename _ForwardIterator , typename _Tp >
  void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _Tp >
  void **fill** (const _Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _Tp &__value)
- template<typename _OIter , typename _Size , typename _Tp >
  _OIter **fill_n** (_OIter, _Size, const _Tp &)
- template<typename _OI , typename _Size , typename _Tp >
  _OI fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  istreambuf_iterator< _CharT >
  >::__type **find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _IIter , typename _Tp >
  _IIter **find** (_IIter, _IIter, const _Tp &)

- template<typename _InputIterator , typename _Tp >
_InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _FIter1 , typename _FIter2 >
_FIter1 **find_end** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
_FIter1 **find_end** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
_ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _FIter1 , typename _FIter2 >
_FIter1 **find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
_FIter1 **find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _InputIterator , typename _ForwardIterator >
_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)
- template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
_InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _IIter , typename _Predicate >
_IIter **find_if** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
_InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _IIter , typename _Predicate >
_IIter **find_if_not** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
_InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- ios_base & fixed (ios_base &__base)
- bitset< _Nb > & flip () noexcept
- bitset< _Nb > & flip (size_t __position)
- constexpr float **floor** (float __x)
- constexpr long double **floor** (long double __x)
- template<typename _Tp >
constexpr
__gnu_cxx::__enable_if
< __is_integer< _Tp >::__value,
double >::__type **floor** (_Tp __x)
- template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits > &__os)
- constexpr float **fmod** (float __x, float __y)
- constexpr long double **fmod** (long double __x, long double __y)
- template<typename _Tp , typename _Up >
constexpr
__gnu_cxx::__promote_2< _Tp,
_Up >::__type **fmod** (_Tp __x, _Up __y)
- template<typename _IIter , typename _Funct >
_Funct **for_each** (_IIter, _IIter, _Funct)
- template<typename _InputIterator , typename _Function >
_Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)

- template<typename _Tp >
  constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename _Tp >
  constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename... _Elements>
  tuple< _Elements &&...> **forward_as_tuple** (_Elements &&...__args) noexcept
- float **frexp** (float __x, int ∗__exp)
- long double **frexp** (long double __x, int ∗__exp)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **frexp** (_Tp __x, int ∗__exp)
- template<typename _Container >
  front_insert_iterator< _Container > front_inserter (_Container &__x)
- const error_category & future_category () noexcept
- template<typename _FIter , typename _Generator >
  void **generate** (_FIter, _FIter, _Generator)
- template<typename _ForwardIterator , typename _Generator >
  void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator >
  _RealType generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **generate_n** (_OIter, _Size, _Generator)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)
- const error_category & **generic_category** () noexcept
- template<std::size_t _Int, class _Tp1 , class _Tp2 >
  constexpr tuple_element< _Int,
  std::pair< _Tp1, _Tp2 >
  >::type & **get** (std::pair< _Tp1, _Tp2 > &__in) noexcept
- template<std::size_t _Int, class _Tp1 , class _Tp2 >
  constexpr tuple_element< _Int,
  std::pair< _Tp1, _Tp2 >
  >::type && **get** (std::pair< _Tp1, _Tp2 > &&__in) noexcept
- template<std::size_t _Int, class _Tp1 , class _Tp2 >
  constexpr const tuple_element
  < _Int, std::pair< _Tp1, _Tp2 >
  >::type & **get** (const std::pair< _Tp1, _Tp2 > &__in) noexcept
- template<std::size_t _Int, typename _Tp , std::size_t _Nm>
  constexpr _Tp & **get** (array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t _Int, typename _Tp , std::size_t _Nm>
  constexpr _Tp && **get** (array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp , std::size_t _Nm>
  constexpr const _Tp & **get** (const array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_ref< typename
  tuple_element< __i, tuple
  < _Elements...> >::type >
  ::type **get** (tuple< _Elements...> &__t) noexcept

- template<std::size_t __i, typename... _Elements>
  constexpr __add_c_ref
  < typename tuple_element< __i,
  tuple< _Elements...> >::type >
  ::type **get** (const tuple< _Elements...> &__t) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_r_ref
  < typename tuple_element< __i,
  tuple< _Elements...> >::type >
  ::type **get** (tuple< _Elements...> &&__t) noexcept
- template<typename _Del , typename _Tp , _Lock_policy _Lp>
  _Del ∗ get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _MoneyT >
  _Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)
- template<typename _Tp >
  pair< _Tp ∗, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<>
  basic_istream< char > & **getline** (basic_istream< char > &__in, basic_string< char > &__str, char __delim)
- template<>
  basic_istream< wchar_t > & **getline** (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)
- template<typename _Facet >
  bool has_facet (const locale &__loc) throw ()
- ios_base & hex (ios_base &__base)
- template<typename _Tp >
  constexpr _Tp **imag** (const complex< _Tp > &__z)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **imag** (_Tp)
- template<typename _IIter1 , typename _IIter2 >
  bool **includes** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _Compare >
  bool **includes** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _InputIterator1 , typename _InputIterator2 >
  bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare >
  bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp >
  _Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)

- template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp , typename _BinaryOperation1 , typename _BinaryOperation2 >
  _Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)
- template<typename _BIter >
  void **inplace_merge** (_BIter, _BIter, _BIter)
- template<typename _BIter , typename _Compare >
  void **inplace_merge** (_BIter, _BIter, _BIter, _Compare)
- template<typename _BidirectionalIterator >
  void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _Container , typename _Iterator >
  insert_iterator< _Container > inserter (_Container &__x, _Iterator __i)
- ios_base & internal (ios_base &__base)
- template<typename _ForwardIterator , typename _Tp >
  void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template<typename _RAIter >
  bool **is_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  bool **is_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter >
  _RAIter **is_heap_until** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  _RAIter **is_heap_until** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _IIter , typename _Predicate >
  bool **is_partitioned** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
  bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _FIter1 , typename _FIter2 >
  bool **is_permutation** (_FIter1, _FIter1, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
  bool **is_permutation** (_FIter1, _FIter1, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _FIter >
  bool **is_sorted** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  bool **is_sorted** (_FIter, _FIter, _Compare)

- template<typename _ForwardIterator >
  bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _FIter >
  _FIter **is_sorted_until** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  _FIter **is_sorted_until** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator >
  _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _CharT >
  bool isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool islower (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isprint (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isspace (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isupper (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool isxdigit (_CharT __c, const locale &__loc)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _FIter1 , typename _FIter2 >
  void **iter_swap** (_FIter1, _FIter2)
- template<typename _Tp >
  _Tp kill_dependency (_Tp __y) noexcept
- constexpr float **ldexp** (float __x, int __exp)
- constexpr long double **ldexp** (long double __x, int __exp)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **ldexp** (_Tp __x, int __exp)
- ios_base & left (ios_base &__base)
- template<typename _IIter1 , typename _IIter2 >
  bool **lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _Compare >
  bool **lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)

- template<typename _II1 , typename _II2 >
  bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1 , typename _II2 , typename _Compare >
  bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _L1 , typename _L2 , typename... _L3>
  void lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)
- template<typename _Tp >
  complex< _Tp > log (const complex< _Tp > &)
- constexpr float **log** (float __x)
- constexpr long double **log** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **log** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Log, _Expr,
  _Dom >, typename
  _Dom::value_type > **log** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Log,
  _ValArray, _Tp >, _Tp > **log** (const valarray< _Tp > &__v)
- template<typename _Tp >
  complex< _Tp > log10 (const complex< _Tp > &)
- constexpr float **log10** (float __x)
- constexpr long double **log10** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **log10** (_Tp __x)
- template<class _Dom >
  _Expr< _UnClos< _Log10, _Expr,
  _Dom >, typename
  _Dom::value_type > **log10** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Log10,
  _ValArray, _Tp >, _Tp > **log10** (const valarray< _Tp > &__v)
- template<typename _FIter , typename _Tp >
  _FIter **lower_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  _FIter **lower_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- error_code make_error_code (future_errc __errc) noexcept
- error_code **make_error_code** (errc __e) noexcept
- error_condition make_error_condition (future_errc __errc) noexcept
- error_condition **make_error_condition** (errc __e) noexcept
- template<typename _Ex >
  exception_ptr make_exception_ptr (_Ex __ex) noexcept

- template<typename _RAIter >
  void **make_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **make_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Iterator >
  move_iterator< _Iterator > **make_move_iterator** (_Iterator __i)
- template<class _T1 , class _T2 >
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)
- template<typename _Tp , typename... _Args>
  shared_ptr< _Tp > make_shared (_Args &&...__args)
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type...> **make_tuple** (_Elements &&...__args)
- template<typename _Tp >
  const _Tp & max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  _Tp **max** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  _Tp **max** (initializer_list< _Tp >, _Compare)
- template<typename _FIter >
  _FIter **max_element** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  _FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator >
  _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp , typename _Class >
  _Mem_fn< _Tp _Class::∗ > mem_fn (_Tp _Class::∗__pm) noexcept
- template<typename _Ret , typename _Tp >
  mem_fun_t< _Ret, _Tp > **mem_fun** (_Ret(_Tp::∗__f)())
- template<typename _Ret , typename _Tp , typename _Arg >
  mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun** (_Ret(_Tp::∗__f)(_Arg))
- template<typename _Ret , typename _Tp >
  mem_fun_ref_t< _Ret, _Tp > **mem_fun_ref** (_Ret(_Tp::∗__f)())
- template<typename _Ret , typename _Tp , typename _Arg >
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun_ref** (_Ret(_Tp::∗__f)(_Arg))
- void ∗ **memchr** (void ∗__s, int __c, size_t __n)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)

- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp >
  const _Tp & min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  _Tp **min** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  _Tp **min** (initializer_list< _Tp >, _Compare)
- template<typename _FIter >
  _FIter **min_element** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  _FIter **min_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator >
  _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp >
  pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  pair< _Tp, _Tp > **minmax** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  pair< _Tp, _Tp > **minmax** (initializer_list< _Tp >, _Compare)
- template<typename _FIter >
  pair< _FIter, _FIter > **minmax_element** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  pair< _FIter, _FIter > **minmax_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator >
  pair< _ForwardIterator,
  _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  pair< _ForwardIterator,
  _ForwardIterator > minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1, _IIter1, _IIter2, _BinaryPredicate)
- template<typename _InputIterator1 , typename _InputIterator2 >
  pair< _InputIterator1,
  _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
  pair< _InputIterator1,
  _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- float **modf** (float __x, float ∗__iptr)
- long double **modf** (long double __x, long double ∗__iptr)

- template<typename _Tp >
  constexpr
  std::remove_reference< _Tp >
  ::type && move (_Tp &&__t) noexcept
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **move** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _II , typename _OI >
  _OI move (_II __first, _II __last, _OI __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **move** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **move_backward** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _BI1 , typename _BI2 >
  _BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp
  &, _Tp ∗ > **move_backward** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  constexpr conditional
  < __move_if_noexcept_cond< _Tp >
  ::value, const _Tp &, _Tp && >
  ::type move_if_noexcept (_Tp &__x) noexcept
- template<typename _ForwardIterator >
  _ForwardIterator **next** (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type __n=1)
- template<typename _BIter >
  bool **next_permutation** (_BIter, _BIter)
- template<typename _BIter , typename _Compare >
  bool **next_permutation** (_BIter, _BIter, _Compare)
- template<typename _BidirectionalIterator >
  bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- ios_base & noboolalpha (ios_base &__base)
- bool none () const noexcept
- template<typename _IIter , typename _Predicate >
  bool **none_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator , typename _Predicate >
  bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp >
  _Tp norm (const complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **norm** (_Tp __x)
- ios_base & noshowbase (ios_base &__base)
- ios_base & noshowpoint (ios_base &__base)
- ios_base & noshowpos (ios_base &__base)

- ios_base & noskipws (ios_base &__base)
- template<typename _Predicate >
  unary_negate< _Predicate > not1 (const _Predicate &__pred)
- template<typename _Predicate >
  binary_negate< _Predicate > not2 (const _Predicate &__pred)
- ios_base & nounitbuf (ios_base &__base)
- ios_base & nouppercase (ios_base &__base)
- template<typename _RAIter >
  void **nth_element** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **nth_element** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)
- ios_base & oct (ios_base &__base)
- template<class _Tp , class _CharT , class _Traits , class _Dist >
  bool operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)
- template<typename _T1 , typename _T2 >
  bool **operator!=** (const allocator< _T1 > &, const allocator< _T2 > &)
- template<typename _Tp >
  bool **operator!=** (const allocator< _Tp > &, const allocator< _Tp > &)
- bool **operator!=** (thread::id __x, thread::id __y) noexcept
- template<typename _CharT , typename _Traits >
  bool **operator!=** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _StateT >
  bool **operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<class _T1 , class _T2 >
  constexpr bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename _Tp , std::size_t _Nm>
  bool **operator!=** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **operator!=** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _Seq >
  bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **operator!=** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp >
  bool operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)
- template<typename _Val >
  bool **operator!=** (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)
- bool **operator!=** (const error_code &__lhs, const error_code &__rhs) noexcept
- bool **operator!=** (const error_code &__lhs, const error_condition &__rhs) noexcept
- template<typename _Tp , typename _Seq >
  bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- bool **operator!=** (const error_condition &__lhs, const error_code &__rhs) noexcept

- bool **operator!=** (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _Iterator >
  bool **operator!=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _Val >
  bool **operator!=** (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _IteratorL , typename _IteratorR >
  bool **operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp >
  bool **operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m>
  bool operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos
  < __not_equal_to, _Expr, _Expr,
  _Dom1, _Dom2 >, typename __fun
  < __not_equal_to, typename
  _Dom1::value_type >
  ::result_type > **operator!=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __not_equal_to,
  typename _Dom::value_type >
  ::result_type > **operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _Constant,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __not_equal_to, typename
  _Dom::value_type >
  ::result_type > **operator!=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __not_equal_to,
  typename _Dom::value_type >
  ::result_type > **operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __not_equal_to, typename
  _Dom::value_type >
  ::result_type > **operator!=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **operator!=** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **operator!=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **operator!=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp , typename _Dp >
  bool **operator!=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
  bool operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **operator!=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename _UIntType , size_t __w, size_t __s, size_t __r>
  bool operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)
- template<typename _BiIter >
  bool operator!= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **operator!=** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator!=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)

- template<typename _Iterator >
  bool **operator!=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _RandomNumberEngine , size_t __p, size_t __r>
  bool operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **operator!=** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator!=** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator!=** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Bi_iter >
  bool operator!= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **operator!=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _Constant, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **operator!=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _Constant,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **operator!=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Bi_iter >
  bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
  bool operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **operator!=** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **operator!=** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Bi_iter >
  bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)

- template<typename _Tp , typename _Alloc >
  bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **operator!=** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **operator!=** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _RandomNumberEngine , size_t __k>
  bool operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _IntType >
  bool operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)
- template<typename _Bi_iter , class _Alloc >
  bool operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _Tp , typename _Alloc >
  bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _IntType >
  bool operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator!= (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _Res , typename... _Args>
  bool operator!= (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _Res , typename... _Args>
  bool operator!= (nullptr_t, const function< _Res(_Args...)> &__f) noexcept
- template<typename _RealType >
  bool operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)

- template<typename _RealType >
  bool operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)
- bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)
- template<typename _IntType >
  bool operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)
- template<typename _IntType >
  bool operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)
- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __modulus,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __modulus,
  typename _Dom1::value_type >
  ::result_type > **operator%** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **operator%** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **operator%** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **operator%** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **operator%** (const valarray< _Tp > &__v, const _Tp &__t)
- constexpr _Ios_Fmtflags **operator&** (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- constexpr memory_order **operator&** (memory_order __m, __memory_order_modifier __mod)
- constexpr _Ios_Openmode **operator&** (_Ios_Openmode __a, _Ios_Openmode __b)
- constexpr launch **operator&** (launch __x, launch __y)
- constexpr _Ios_Iostate **operator&** (_Ios_Iostate __a, _Ios_Iostate __b)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __bitwise_and,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __bitwise_and, typename
  _Dom1::value_type >
  ::result_type > **operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **operator&** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **operator&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **operator&** (const valarray< _Tp > &__v, const _Tp &__t)

- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __logical_and,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __logical_and, typename
  _Dom1::value_type >
  ::result_type > **operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **operator&&** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >

_Expr< _BinClos< __logical_and,
_Constant, _ValArray, _Tp, _Tp >
, typename __fun
< __logical_and, _Tp >
::result_type > **operator&&** (const _Tp &__t, const valarray< _Tp > &__v)

- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **operator&&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- const _Ios_Fmtflags & **operator&=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- const _Ios_Openmode & **operator&=** (_Ios_Openmode &__a, _Ios_Openmode __b)
- launch & **operator&=** (launch &__x, launch __y)
- const _Ios_Iostate & **operator&=** (_Ios_Iostate &__a, _Ios_Iostate __b)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **operator∗** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **operator∗** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __multiplies,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __multiplies,
  typename _Dom1::value_type >
  ::result_type > **operator∗** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **operator∗** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **operator∗** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **operator*** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **operator*** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **operator*** (const _Tp &__t, const valarray< _Tp > &__v)
- _Bit_iterator **operator+** (ptrdiff_t __n, const _Bit_iterator &__x)
- template<typename _Iterator >
  reverse_iterator< _Iterator > **operator+** (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _Tp , typename _Ref , typename _Ptr >
  _Deque_iterator< _Tp, _Ref, _Ptr > **operator+** (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr > &__x)
- _Bit_const_iterator **operator+** (ptrdiff_t __n, const _Bit_const_iterator &__x)
- template<class _Dom >
  _Expr< _BinClos< __plus, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __plus, typename
  _Dom::value_type >
  ::result_type > **operator+** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __plus, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename __fun< __plus,
  typename _Dom1::value_type >
  ::result_type > **operator+** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __plus,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __plus,
  typename _Dom::value_type >
  ::result_type > **operator+** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __plus, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __plus, typename
  _Dom::value_type >
  ::result_type > **operator+** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __plus,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __plus,
  typename _Dom::value_type >
  ::result_type > **operator+** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  complex< _Tp > operator+ (const complex< _Tp > &__x)
- template<typename _Iterator >
  move_iterator< _Iterator > **operator+** (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **operator+** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **operator+** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **operator+** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > operator+ (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > **operator+** (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits,
_Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits,
_Alloc > **operator+** (const _CharT ∗__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits,
_Alloc > **operator+** (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits,
_Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits,
_Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)
- ptrdiff_t **operator-** (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)
- template<typename _Iterator >
reverse_iterator< _Iterator >
::difference_type **operator-** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
_Deque_iterator< _Tp, _Ref,
_Ptr >::difference_type **operator-** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
_Deque_iterator< _Tp, _RefL,
_PtrL >::difference_type **operator-** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _IteratorL , typename _IteratorR >
auto **operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base()-__x.base())
- template<class _Dom >
_Expr< _BinClos< __minus,
_Expr, _ValArray, _Dom,
typename _Dom::value_type >
, typename __fun< __minus,
typename _Dom::value_type >
::result_type > **operator-** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __minus,
_Expr, _Constant, _Dom,
typename _Dom::value_type >
, typename __fun< __minus,
typename _Dom::value_type >
::result_type > **operator-** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
_Expr< _BinClos< __minus,
_Constant, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __minus,
typename _Dom::value_type >
::result_type > **operator-** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value-

_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __minus,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __minus,
  typename _Dom::value_type >
  ::result_type > **operator-** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __minus,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __minus,
  typename _Dom1::value_type >
  ::result_type > **operator-** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _Tp >
  complex< _Tp > operator- (const complex< _Tp > &__x)
- template<typename _IteratorL , typename _IteratorR >
  auto **operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(-__x.base()-__y.base())
- template<typename _Iterator >
  auto **operator-** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) -> decltype(__x.base()-__y.base())
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **operator-** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **operator-** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **operator-** (const _Tp &__t, const valarray< _Tp > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __divides,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __divides,
  typename _Dom1::value_type >
  ::result_type > **operator/** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __divides,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **operator/** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __divides,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **operator/** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __divides,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **operator/** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __divides,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **operator/** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **operator/** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **operator/** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **operator/** (const _Tp &__t, const valarray< _Tp > &__v)
- bool **operator<** (const error_code &__lhs, const error_code &__rhs) noexcept
- template<class _T1 , class _T2 >
  constexpr bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename _Tp , std::size_t _Nm>
  bool **operator<** (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)
- bool **operator<** (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _Tp , typename _Seq >
  bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **operator<** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **operator<** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)

- template<typename _Tp , typename _Seq >
  bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Iterator >
  bool **operator**< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator**< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator**< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **operator**< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **operator**< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<class _Dom >
  _Expr< _BinClos< __less, _ValArray, _Expr, typename _Dom::value_type, _Dom > , typename __fun< __less, typename _Dom::value_type > ::result_type > **operator**< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __less, _Expr, _Expr, _Dom1, _Dom2 > , typename __fun< __less, typename _Dom1::value_type > ::result_type > **operator**< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type > ::result_type > **operator**< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __less, _Constant, _Expr, typename _Dom::value_type, _Dom > , typename __fun< __less, typename _Dom::value_type > ::result_type > **operator**< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< __less, typename _Dom::value_type > ::result_type > **operator**< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **operator**< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)

- template<typename _Tp , typename _Dp >
  bool **operator**< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **operator**< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **operator**< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename _BiIter >
  bool operator< (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **operator**< (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator**< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **operator**< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **operator**< (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator**< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Bi_iter >
  bool operator< (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator**< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **operator**< (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **operator**< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

- template<typename _Tp >
  _Expr< _BinClos< __less,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **operator**< (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Bi_iter >
  bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator< (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)
- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)

- template<class _CharT , class _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)
- template<typename _CharT , typename _Traits , typename _MoneyT >
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **operator**<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **operator**<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __shift_left,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __shift_left,
  typename _Dom1::value_type >
  ::result_type > **operator**<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _Tp , typename _CharT , class _Traits >
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)

- template<typename _CharT , typename _Traits , typename _Tp >
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &&__os, const _Tp &__x)
- template<typename _UIntType , size_t __w, size_t __s, size_t __r, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template<typename _RandomNumberEngine , size_t __p, size_t __r, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _RandomNumberEngine , size_t __k, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **operator**<< (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **operator**<< (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **operator**<< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _RandomNumberEngine , size_t __w, typename _UIntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)
- template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter >
  basic_ostream< _Ch_type,
  _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)

- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits , typename _Alloc >
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _IntType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)

- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- bool **operator<=** (thread::id __x, thread::id __y) noexcept
- template<class _T1 , class _T2 >
  constexpr bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename _Tp , std::size_t _Nm>
  bool **operator<=** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , typename _Seq >
  bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **operator<=** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _Seq >
  bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **operator<=** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Iterator >
  bool **operator<=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator<=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<class _Dom >
  _Expr< _BinClos< __less_equal,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __less_equal,
  typename _Dom::value_type >
  ::result_type > **operator<=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >

_Expr< _BinClos< __less_equal,
_ValArray, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __less_equal,
typename _Dom::value_type >
::result_type > **operator<=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom1 , class _Dom2 >
_Expr< _BinClos< __less_equal,
_Expr, _Expr, _Dom1, _Dom2 >
, typename __fun< __less_equal,
typename _Dom1::value_type >
::result_type > **operator<=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
_Expr< _BinClos< __less_equal,
_Constant, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __less_equal,
typename _Dom::value_type >
::result_type > **operator<=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __less_equal,
_Expr, _Constant, _Dom,
typename _Dom::value_type >
, typename __fun< __less_equal,
typename _Dom::value_type >
::result_type > **operator<=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool **operator<=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
bool **operator<=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
bool **operator<=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Key , typename _Compare , typename _Alloc >
bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename... _TElements, typename... _UElements>
constexpr bool **operator<=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename _BiIter >
bool operator<= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
bool **operator<=** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)

- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator<=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **operator<=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **operator<=** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator<=** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator<=** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Bi_iter >
  bool operator<= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **operator<=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **operator<=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **operator<=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Bi_iter >
  bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator<= (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _Tp , typename _CharT , typename _Traits , typename _Dist >
  bool operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)
- template<typename _T1 , typename _T2 >
  bool **operator==** (const allocator< _T1 > &, const allocator< _T2 > &)
- template<typename _Tp >
  bool **operator==** (const allocator< _Tp > &, const allocator< _Tp > &)
- template<typename _CharT , typename _Traits >
  bool **operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<class _T1 , class _T2 >
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename _StateT >
  bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _Tp , std::size_t _Nm>
  bool **operator==** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , typename _Seq >
  bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **operator==** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **operator==** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp >
  bool operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)
- bool **operator==** (const error_code &__lhs, const error_code &__rhs) noexcept
- bool **operator==** (const error_code &__lhs, const error_condition &__rhs) noexcept
- template<typename _Tp , typename _Seq >
  bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- bool **operator==** (const error_condition &__lhs, const error_code &__rhs) noexcept
- template<typename _Val >
  bool **operator==** (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)
- bool **operator==** (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _Iterator >
  bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _Val >
  bool **operator==** (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _IteratorL , typename _IteratorR >
  bool **operator==** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)

- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **operator==** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **operator==** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __equal_to,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __equal_to,
  typename _Dom1::value_type >
  ::result_type > **operator==** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **operator==** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **operator==** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **operator==** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **operator==** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp , typename _Dp >
  bool **operator==** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)

- template<typename... _TElements, typename... _UElements>
  constexpr bool **operator==** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename _BiIter >
  bool operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **operator==** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **operator==** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator==** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator==** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Bi_iter >
  bool operator== (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **operator==** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **operator==** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **operator==** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Bi_iter >
  bool operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)

- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Bi_iter >
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Bi_iter , typename _Alloc >
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _Tp , typename _Alloc >
  bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _RealType >
  bool **operator==** (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  bool >::__type **operator==** (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator== (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _Res , typename... _Args>
  bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _Res , typename... _Args>
  bool operator== (nullptr_t, const function< _Res(_Args...)> &__f) noexcept
- bool **operator>** (thread::id __x, thread::id __y) noexcept
- template<class _T1 , class _T2 >
  constexpr bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename _Tp , std::size_t _Nm>
  bool **operator>** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , typename _Seq >
  bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)

- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **operator>** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **operator>** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Seq >
  bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Iterator >
  bool **operator>** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<class _Dom >
  _Expr< _BinClos< __greater,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **operator>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __greater,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **operator>** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __greater,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **operator>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __greater,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __greater,
  typename _Dom1::value_type >
  ::result_type > **operator>** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >

_Expr< _BinClos< __greater,
_Expr, _ValArray, _Dom,
typename _Dom::value_type >
, typename __fun< __greater,
typename _Dom::value_type >
::result_type > **operator**> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool **operator**> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
bool **operator**> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
bool **operator**> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Key , typename _Compare , typename _Alloc >
bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename... _TElements, typename... _UElements>
constexpr bool **operator**> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _BiIter >
bool operator> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
bool **operator**> (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
bool operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
bool **operator**> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
bool **operator**> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Bi_iter >
bool operator> (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
bool **operator**> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
bool **operator**> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
bool **operator**> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept

- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **operator**> (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **operator**> (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **operator**> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Bi_iter >
  bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator> (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- bool **operator**>= (thread::id __x, thread::id __y) noexcept
- template<class _T1 , class _T2 >
  constexpr bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<typename _Tp , std::size_t _Nm>
  bool **operator**>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , typename _Seq >
  bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **operator**>= (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **operator**>= (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)

- template<typename _Iterator >
  bool **operator>=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp1 , typename _Tp2 >
  bool **operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __greater_equal,
  typename _Dom::value_type >
  ::result_type > **operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __greater_equal,
  typename _Dom::value_type >
  ::result_type > **operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _Constant,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __greater_equal, typename
  _Dom::value_type >
  ::result_type > **operator>=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos
  < __greater_equal, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __greater_equal, typename
  _Dom1::value_type >
  ::result_type > **operator>=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __greater_equal, typename
  _Dom::value_type >
  ::result_type > **operator>=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, type-

name _Dom::value_type > &__e)

- template<typename _Tp >
  bool **operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **operator>=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **operator>=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **operator>=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **operator>=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename _BiIter >
  bool operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **operator>=** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **operator>=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Bi_iter >
  bool operator>= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **operator>=** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator>=** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **operator>=** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **operator>=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _Constant, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **operator>=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _Constant,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **operator>=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Bi_iter >
  bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool operator>= (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setbase __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)
- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)

- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **operator**>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)
- template<typename _CharT , typename _Traits , typename _MoneyT >
  basic_istream< _CharT, _Traits > & **operator**>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __shift_right,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __shift_right, typename
  _Dom1::value_type >
  ::result_type > **operator**>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **operator**>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **operator**>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **operator**>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **operator**>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp , typename _CharT , class _Traits >
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT , typename _Traits >

std::basic_istream< _CharT,
_Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)

- template<typename _UIntType , size_t __w, size_t __s, size_t __r, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template<typename _RandomNumberEngine , size_t __p, size_t __r, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _RandomNumberEngine , size_t __k, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT , typename _Traits , typename _Tp >
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &&__is, _Tp &__x)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **operator**>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **operator**>> (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **operator**>> (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)

- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<>
  basic_istream< char > & **operator>>** (basic_istream< char > &__is, basic_string< char > &__str)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)

- template< typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template< typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)
- template< typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- constexpr _Ios_Fmtflags **operator^** (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- constexpr _Ios_Openmode **operator^** (_Ios_Openmode __a, _Ios_Openmode __b)
- constexpr launch **operator^** (launch __x, launch __y)
- constexpr _Ios_Iostate **operator^** (_Ios_Iostate __a, _Ios_Iostate __b)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **operator^** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **operator^** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **operator^** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **operator^** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __bitwise_xor,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __bitwise_xor, typename
  _Dom1::value_type >
  ::result_type > **operator^** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **operator^** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **operator^** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **operator^** (const _Tp &__t, const valarray< _Tp > &__v)
- const _Ios_Fmtflags & **operator^=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- const _Ios_Openmode & **operator^=** (_Ios_Openmode &__a, _Ios_Openmode __b)
- launch & **operator^=** (launch &__x, launch __y)
- const _Ios_Iostate & **operator^=** (_Ios_Iostate &__a, _Ios_Iostate __b)
- bitset< _Nb > & **operator^=** (const bitset< _Nb > &__rhs) noexcept
- constexpr memory_order **operator|** (memory_order __m, __memory_order_modifier __mod)
- constexpr _Ios_Fmtflags **operator|** (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- constexpr _Ios_Openmode **operator|** (_Ios_Openmode __a, _Ios_Openmode __b)
- constexpr launch **operator|** (launch __x, launch __y)
- constexpr _Ios_Iostate **operator|** (_Ios_Iostate __a, _Ios_Iostate __b)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **operator|** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **operator|** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **operator|** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __bitwise_or,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __bitwise_or,
  typename _Dom1::value_type >
  ::result_type > **operator|** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **operator|** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **operator|** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **operator|** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **operator|** (const _Tp &__t, const valarray< _Tp > &__v)
- const _Ios_Fmtflags & **operator|=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- const _Ios_Openmode & **operator|=** (_Ios_Openmode &__a, _Ios_Openmode __b)
- launch & **operator|=** (launch &__x, launch __y)
- const _Ios_Iostate & **operator|=** (_Ios_Iostate &__a, _Ios_Iostate __b)
- bitset< _Nb > & **operator|=** (const bitset< _Nb > &__rhs) noexcept
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __logical_or,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __logical_or,
  typename _Dom1::value_type >
  ::result_type > **operator||** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >

_Expr< _BinClos< __logical_or,
_Expr, _Constant, _Dom,
typename _Dom::value_type >
, typename __fun< __logical_or,
typename _Dom::value_type >
::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom >
_Expr< _BinClos< __logical_or,
_ValArray, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __logical_or,
typename _Dom::value_type >
::result_type > **operator**|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
_Expr< _BinClos< __logical_or,
_Expr, _ValArray, _Dom,
typename _Dom::value_type >
, typename __fun< __logical_or,
typename _Dom::value_type >
::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
_Expr< _BinClos< __logical_or,
_Constant, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __logical_or,
typename _Dom::value_type >
::result_type > **operator**|| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp >
_Expr< _BinClos< __logical_or,
_ValArray, _ValArray, _Tp, _Tp >
, typename __fun< __logical_or,
_Tp >::result_type > **operator**|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
_Expr< _BinClos< __logical_or,
_Constant, _ValArray, _Tp, _Tp >
, typename __fun< __logical_or,
_Tp >::result_type > **operator**|| (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
_Expr< _BinClos< __logical_or,
_ValArray, _Constant, _Tp, _Tp >
, typename __fun< __logical_or,
_Tp >::result_type > **operator**|| (const valarray< _Tp > &__v, const _Tp &__t)
- constexpr _Ios_Fmtflags **operator**∼ (_Ios_Fmtflags __a)
- constexpr _Ios_Openmode **operator**∼ (_Ios_Openmode __a)
- constexpr launch **operator**∼ (launch __x)
- constexpr _Ios_Iostate **operator**∼ (_Ios_Iostate __a)
- bitset< _Nb > operator∼ () const noexcept
- template<typename _RAIter >
void **partial_sort** (_RAIter, _RAIter, _RAIter)

- template<typename _RAIter , typename _Compare >
  void **partial_sort** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _IIter , typename _RAIter >
  _RAIter **partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter)
- template<typename _IIter , typename _RAIter , typename _Compare >
  _RAIter **partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter, _Compare)
- template<typename _InputIterator , typename _RandomAccessIterator >
  _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _BIter , typename _Predicate >
  _BIter **partition** (_BIter, _BIter, _Predicate)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _IIter , typename _OIter1 , typename _OIter2 , typename _Predicate >
  pair< _OIter1, _OIter2 > **partition_copy** (_IIter, _IIter, _OIter1, _OIter2, _Predicate)
- template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate >
  pair< _OutputIterator1,
  _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _FIter , typename _Predicate >
  _FIter **partition_point** (_FIter, _FIter, _Predicate)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _Tp >
  complex< _Tp > polar (const _Tp &, const _Tp &=0)
- template<typename _RandomAccessIterator >
  void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter >
  void **pop_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **pop_heap** (_RAIter, _RAIter, _Compare)
- template<typename _Tp >
  complex< _Tp > pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp >
  complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > pow (const _Tp &, const complex< _Tp > &)

- constexpr float **pow** (float __x, float __y)
- constexpr long double **pow** (long double __x, long double __y)
- template<typename _Tp , typename _Up >
  constexpr
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **pow** (_Tp __x, _Up __y)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< _Pow, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename _Dom1::value_type > **pow** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom >
  _Expr< _BinClos< _Pow,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _BinClos< _Pow,
  _ValArray, _Constant, _Tp, _Tp >
  , _Tp > **pow** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< _Pow,
  _Constant, _ValArray, _Tp, _Tp >
  , _Tp > **pow** (const _Tp &__t, const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _BinClos< _Pow,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _BinClos< _Pow,
  _ValArray, _ValArray, _Tp, _Tp >
  , _Tp > **pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom >
  _Expr< _BinClos< _Pow, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< _Pow, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **pow** (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,

_Up >::__type > **pow** (const _Tp &__x, const std::complex< _Up > &__y)

- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _BidirectionalIterator >
  _BidirectionalIterator **prev** (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)
- template<typename _BIter >
  bool **prev_permutation** (_BIter, _BIter)
- template<typename _BIter , typename _Compare >
  bool **prev_permutation** (_BIter, _BIter, _Compare)
- template<typename _BidirectionalIterator >
  bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _Tp >
  std::complex< _Tp > **proj** (const std::complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **proj** (_Tp __x)
- template<typename _Arg , typename _Result >
  pointer_to_unary_function
  < _Arg, _Result > ptr_fun (_Result(∗__x)(_Arg))
- template<typename _Arg1 , typename _Arg2 , typename _Result >
  pointer_to_binary_function
  < _Arg1, _Arg2, _Result > ptr_fun (_Result(∗__x)(_Arg1, _Arg2))
- template<typename _RandomAccessIterator >
  void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter >
  void **push_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **push_heap** (_RAIter, _RAIter, _Compare)
- template<typename _MoneyT >
  _Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)
- template<typename _RAIter >
  void **random_shuffle** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Generator >
  void **random_shuffle** (_RAIter, _RAIter, _Generator &&)
- template<typename _RandomAccessIterator >
  void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _RandomNumberGenerator >
  void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)
- template<typename _Tp >
  constexpr _Tp **real** (const complex< _Tp > &__z)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **real** (_Tp __x)
- template<typename _FIter , typename _Tp >
  _FIter **remove** (_FIter, _FIter, const _Tp &)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- template<typename _IIter , typename _OIter , typename _Tp >
  _OIter **remove_copy** (_IIter, _IIter, _OIter, const _Tp &)
- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **remove_copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
  _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _FIter , typename _Predicate >
  _FIter **remove_if** (_FIter, _FIter, _Predicate)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _FIter , typename _Tp >
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _ForwardIterator , typename _Tp >
  void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _IIter , typename _OIter , typename _Tp >
  _OIter **replace_copy** (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)
- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _Iter , typename _OIter , typename _Predicate , typename _Tp >
  _OIter **replace_copy_if** (_Iter, _Iter, _OIter, _Predicate, const _Tp &)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp >
  _OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _ForwardIterator , typename _Predicate , typename _Tp >
  void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- bitset< _Nb > & reset () noexcept
- bitset< _Nb > & reset (size_t __position)
- _Resetiosflags resetiosflags (ios_base::fmtflags __mask)
- void rethrow_exception (exception_ptr) __attribute__((__noreturn__))
- template<typename _Ex >
  void rethrow_if_nested (const _Ex &__ex)
- void rethrow_if_nested (const nested_exception &__ex)
- template<typename _Tp >
  void return_temporary_buffer (_Tp *__p)
- template<typename _BIter >
  void **reverse** (_BIter, _BIter)
- template<typename _BidirectionalIterator >
  void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BIter , typename _OIter >
  _OIter **reverse_copy** (_BIter, _BIter, _OIter)
- template<typename _BidirectionalIterator , typename _OutputIterator >
  _OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- ios_base & right (ios_base &__base)
- template<typename _FIter >
  void **rotate** (_FIter, _FIter, _FIter)

- template<typename _ForwardIterator >
  void rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _FIter , typename _OIter >
  _OIter **rotate_copy** (_FIter, _FIter, _FIter, _OIter)
- template<typename _ForwardIterator , typename _OutputIterator >
  _OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- ios_base & scientific (ios_base &__base)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _FIter , typename _Size , typename _Tp >
  _FIter **search_n** (_FIter, _FIter, _Size, const _Tp &)
- template<typename _FIter , typename _Size , typename _Tp , typename _BinaryPredicate >
  _FIter **search_n** (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)
- template<typename _ForwardIterator , typename _Integer , typename _Tp >
  _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- bitset< _Nb > & set () noexcept
- bitset< _Nb > & set (size_t __position, bool __val=true)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- new_handler set_new_handler (new_handler) throw ()
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)

- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- terminate_handler set_terminate (terminate_handler) noexcept
- unexpected_handler set_unexpected (unexpected_handler) noexcept
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- _Setbase setbase (int __base)
- template<typename _CharT >
  _Setfill< _CharT > setfill (_CharT __c)
- _Setiosflags setiosflags (ios_base::fmtflags __mask)
- _Setprecision setprecision (int __n)
- _Setw setw (int __n)
- ios_base & showbase (ios_base &__base)
- ios_base & showpoint (ios_base &__base)
- ios_base & showpos (ios_base &__base)
- template<typename _RAIter , typename _UGenerator >
  void **shuffle** (_RAIter, _RAIter, _UGenerator &&)
- template<typename _RandomAccessIterator , typename _UniformRandomNumberGenerator >
  void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)
- template<typename _Tp >
  complex< _Tp > sin (const complex< _Tp > &)
- constexpr float **sin** (float __x)
- template<class _Dom >
  _Expr< _UnClos< _Sin, _Expr,
  _Dom >, typename
  _Dom::value_type > **sin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Sin,
  _ValArray, _Tp >, _Tp > **sin** (const valarray< _Tp > &__v)
- constexpr long double **sin** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **sin** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > sinh (const complex< _Tp > &)

- template<typename _Tp >
  _Expr< _UnClos< _Sinh,
  _ValArray, _Tp >, _Tp > **sinh** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Sinh, _Expr,
  _Dom >, typename
  _Dom::value_type > **sinh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **sinh** (float __x)
- constexpr long double **sinh** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **sinh** (_Tp __x)
- constexpr size_t size () const noexcept
- ios_base & skipws (ios_base &__base)
- template<typename _RAIter >
  void **sort** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **sort** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter >
  void **sort_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **sort_heap** (_RAIter, _RAIter, _Compare)
- template<typename _Tp >
  complex< _Tp > sqrt (const complex< _Tp > &)
- template<typename _Tp >
  _Expr< _UnClos< _Sqrt,
  _ValArray, _Tp >, _Tp > **sqrt** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Sqrt, _Expr,
  _Dom >, typename
  _Dom::value_type > **sqrt** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **sqrt** (float __x)
- constexpr long double **sqrt** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **sqrt** (_Tp __x)
- template<typename _BIter , typename _Predicate >
  _BIter **stable_partition** (_BIter, _BIter, _Predicate)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)

- template<typename _RAIter >
  void **stable_sort** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **stable_sort** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator >
  void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- char ∗ **strchr** (char ∗__s, int __n)
- char ∗ **strpbrk** (char ∗__s1, const char ∗__s2)
- char ∗ **strrchr** (char ∗__s, int __n)
- char ∗ **strstr** (char ∗__s1, const char ∗__s2)
- void **swap** (_Bit_reference __x, _Bit_reference __y) noexcept
- void **swap** (_Bit_reference __x, bool &__y) noexcept
- void **swap** (bool &__x, _Bit_reference __y) noexcept
- void **swap** (thread &__x, thread &__y) noexcept
- template<class _T1 , class _T2 >
  void swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp , std::size_t _Nm>
  void **swap** (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))
- template<typename _Tp , typename _Seq >
  void **swap** (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp , typename _Seq >
  void **swap** (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp >
  void **swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp , typename _Dp >
  void **swap** (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- template<typename _Tp >
  void **swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _Tp >
  void swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >
- template<typename _Tp , typename _Sequence , typename _Compare >
  void **swap** (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp , size_t _Nm>
  void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(∗__a
- template<typename _Mutex >
  void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept
- template<typename _Ch_type , typename _Rx_traits >
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)
- template<typename _Key , typename _Compare , typename _Alloc >
  void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)

- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  void **swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename... _Elements>
  void swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp , _Lock_policy _Lp>
  void **swap** (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  void **swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , _Lock_policy _Lp>
  void **swap** (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)
- template<typename _Bi_iter , typename _Alloc >
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)
- template<typename _Res , typename... _Args>
  void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)
- template<typename _CharT , typename _Traits , typename _Alloc >
  void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _FIter1 , typename _FIter2 >
  _FIter2 **swap_ranges** (_FIter1, _FIter1, _FIter2)
- const error_category & **system_category** () noexcept
- template<typename _Tp >
  complex< _Tp > tan (const complex< _Tp > &)
- template<typename _Tp >
  _Expr< _UnClos< _Tan,
  _ValArray, _Tp >, _Tp > **tan** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Tan, _Expr,
  _Dom >, typename
  _Dom::value_type > **tan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **tan** (float __x)

- constexpr long double **tan** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **tan** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > tanh (const complex< _Tp > &)
- template<typename _Tp >
  _Expr< _UnClos< _Tanh,
  _ValArray, _Tp >, _Tp > **tanh** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Tanh, _Expr,
  _Dom >, typename
  _Dom::value_type > **tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **tanh** (float __x)
- constexpr long double **tanh** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **tanh** (_Tp __x)
- void terminate () noexcept __attribute__((__noreturn__))
- bool test (size_t __position) const
- template<typename _Ex >
  void throw_with_nested (_Ex __ex)
- template<typename... _Elements>
  tuple< _Elements &...> tie (_Elements &...__args) noexcept
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_string< _CharT,
  _Traits, _Alloc > to_string () const
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_string< _CharT,
  _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT , class _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **to_string** () const
- template<class _CharT , class _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **to_string** () const
- template<class _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- std::basic_string< char,
  std::char_traits< char >
  , std::allocator< char > > **to_string** (char __zero, char __one= '1') const

- unsigned long long **to_ullong** () const
- unsigned long to_ulong () const
- template<typename _CharT >
  _CharT tolower (_CharT __c, const locale &__loc)
- template<typename _CharT >
  _CharT toupper (_CharT __c, const locale &__loc)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BinaryOperation >
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)
- template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
  int try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto tuple_cat (_Tpls &&...__tpls) -> typename __tuple_cat_result< _Tpls...>::__type
- bool uncaught_exception () noexcept __attribute__((__pure__))
- void unexpected () __attribute__((__noreturn__))
- template<typename _InputIterator , typename _ForwardIterator >
  _ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator , typename _Size , typename _ForwardIterator >
  _ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator , typename _Tp >
  void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator , typename _Size , typename _Tp >
  void uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _FIter >
  _FIter **unique** (_FIter, _FIter)
- template<typename _FIter , typename _BinaryPredicate >
  _FIter **unique** (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator >
  _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _IIter , typename _OIter >
  _OIter **unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter , typename _OIter , typename _BinaryPredicate >
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- ios_base & unitbuf (ios_base &__base)
- template<typename _FIter , typename _Tp >
  _FIter **upper_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  _FIter **upper_bound** (_FIter, _FIter, const _Tp &, _Compare)

- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- ios_base & uppercase (ios_base &__base)
- template<typename _Facet >
  const _Facet & use_facet (const locale &__loc)
- wchar_t ∗ **wcschr** (wchar_t ∗__p, wchar_t __c)
- wchar_t ∗ **wcspbrk** (wchar_t ∗__s1, const wchar_t ∗__s2)
- wchar_t ∗ **wcsrchr** (wchar_t ∗__p, wchar_t __c)
- wchar_t ∗ **wcsstr** (wchar_t ∗__s1, const wchar_t ∗__s2)
- wchar_t ∗ **wmemchr** (wchar_t ∗__p, wchar_t __c, size_t __n)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > &__is)

- bitset< _Nb > & operator<<= (size_t __position) noexcept
- bitset< _Nb > & operator>>= (size_t __position) noexcept

- bitset< _Nb > & _Unchecked_set (size_t __pos) noexcept
- bitset< _Nb > & _Unchecked_set (size_t __pos, int __val) noexcept
- bitset< _Nb > & _Unchecked_reset (size_t __pos) noexcept
- bitset< _Nb > & _Unchecked_flip (size_t __pos) noexcept
- constexpr bool _Unchecked_test (size_t __pos) const noexcept

- reference operator[] (size_t __position)

- bool operator== (const bitset< _Nb > &__rhs) const noexcept
- bool operator!= (const bitset< _Nb > &__rhs) const noexcept

- bitset< _Nb > operator<< (size_t __position) const noexcept
- bitset< _Nb > operator>> (size_t __position) const noexcept

- template<size_t _Nb>
  bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

- template<class _CharT , class _Traits , size_t _Nb>
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<class _CharT , class _Traits , size_t _Nb>
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)

- template<typename _Tp >
  complex< _Tp > operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > operator+ (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _Tp >
  complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  constexpr bool operator== (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  constexpr bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  constexpr bool operator!= (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  constexpr bool operator!= (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  reference_wrapper< _Tp > ref (_Tp &__t) noexcept
- template<typename _Tp >
  reference_wrapper< const _Tp > cref (const _Tp &__t) noexcept
- template<typename _Tp >
  void ref (const _Tp &&)=delete
- template<typename _Tp >
  void cref (const _Tp &&)=delete
- template<typename _Tp >
  reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t) noexcept
- template<typename _Tp >
  reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t) noexcept

- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<class _Traits >
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)
- template<class _Traits >
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)

- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT ∗__s)
- template<>
  basic_istream< char > & operator>> (basic_istream< char > &__in, char ∗__s)
- template<class _Traits >
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char ∗__s)
- template<class _Traits >
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char ∗__s)

- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)
- template<class _Traits >
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)
- template<class _Traits >
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)
- template<class _Traits >
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)

- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT ∗__s)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char ∗__s)
- template<class _Traits >
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char ∗__s)
- template<class _Traits >
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char ∗__s)
- template<class _Traits >
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char ∗__s)

Matching, Searching, and Replacing

- template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
  bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , typename _Alloc , typename _Rx_traits >
  bool regex_match (const _Ch_type ∗__s, match_results< const _Ch_type ∗, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)

- template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , class _Rx_traits >
  bool regex_match (const _Ch_type ∗__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _Str_allocator , typename _Ch_type , typename _Rx_traits >
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
  bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , class _Alloc , class _Rx_traits >
  bool regex_search (const _Ch_type ∗__s, match_results< const _Ch_type ∗, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type , typename _Rx_traits >
  bool regex_search (const _Ch_type ∗__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_traits >
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type >
  _Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits , typename _Ch_type >
  basic_string< _Ch_type > regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

Variables

- **__a**
- void ∗ **__b**
- static ios_base::Init __ioinit
- function< void()> **__once_functor**
- void
  is_nothrow_move_assignable
  < _Tp >::value _Tp **__tmp**
- constexpr adopt_lock_t **adopt_lock**
- constexpr allocator_arg_t **allocator_arg**
- constexpr defer_lock_t **defer_lock**
- const _Swallow_assign **ignore**
- error_code **make_error_code** (errc) noexcept

- error_condition **make_error_condition** (errc) noexcept
- const nothrow_t **nothrow**
- decltype(nullptr) typedef **nullptr_t**
- constexpr piecewise_construct_t piecewise_construct
- return ∗ **this**
- constexpr try_to_lock_t **try_to_lock**

Standard Stream Objects

The <iostream> header declares the eight standard stream objects. For other declarations, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html` *and the I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` *streams, and to be available during program startup and termination. For more information, see the HOWTO linked to above.*

- istream cin
- ostream cout
- ostream cerr
- ostream clog
- wistream wcin
- wostream wcout
- wostream wcerr
- wostream wclog

3.11.1 Detailed Description

ISO C++ entities toplevel namespace is std.

3.11.2 Typedef Documentation

3.11.2.1 template<bool _Cache> using std::__umap_traits = typedef __detail::_Hashtable_traits<_Cache, false, true>

Base types for unordered_map.

Definition at line 39 of file unordered_map.h.

3.11.2.2 template<bool _Cache> using std::__ummap_traits = typedef __detail::_Hashtable_traits<_Cache, false, false>

Base types for unordered_multimap.

Definition at line 56 of file unordered_map.h.

3.11.2.3 template<bool _Cache> using std::__umset_traits = typedef __detail::_Hashtable_traits<_Cache, true, false>

Base types for unordered_multiset.

Definition at line 54 of file unordered_set.h.

3.11.2.4 template<bool _Cache> using std::__uset_traits = typedef __detail::_Hashtable_traits<_Cache, true, true>

Base types for unordered_set.

Definition at line 39 of file unordered_set.h.

3.11.2.5 typedef void(∗ std::new_handler)()

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 73 of file new.

3.11.2.6 typedef long long **std::streamoff**

Type used by fpos, char_traits<char>, and char_traits<wchar_t>.

In clauses 21.1.3.1 and 27.4.1 streamoff is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, streamoff was typedef long.

Definition at line 94 of file postypes.h.

3.11.2.7 typedef **fpos**<mbstate_t> **std::streampos**

File position for char streams.

Definition at line 228 of file postypes.h.

3.11.2.8 typedef ptrdiff_t **std::streamsize**

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file postypes.h.

3.11.2.9 typedef **fpos**<mbstate_t> **std::u16streampos**

File position for char16_t streams.

Definition at line 234 of file postypes.h.

3.11.2.10 typedef **fpos**<mbstate_t> **std::u32streampos**

File position for char32_t streams.

Definition at line 236 of file postypes.h.

3.11.2.11 typedef **fpos**<mbstate_t> **std::wstreampos**

File position for wchar_t streams.

Definition at line 230 of file postypes.h.

3.11.3 Enumeration Type Documentation

3.11.3.1 anonymous enum

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html` This controls some aspect of the sort routines.

Definition at line 2225 of file stl_algo.h.

3.11.3.2 enum **std::float_denorm_style**

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the std::numeric_limits class.

Enumerator:

denorm_indeterminate	Indeterminate at compile time whether denormalized values are allowed.
denorm_absent	The type does not allow denormalized values.
denorm_present	The type allows denormalized values.

Definition at line 171 of file limits.

3.11.3.3 enum std::float_round_style

Describes the rounding style for floating-point types.

This is used in the std::numeric_limits class.

Enumerator:

round_toward_zero Intermediate.

round_to_nearest To zero.

round_toward_infinity To the nearest representable value.

round_toward_neg_infinity To infinity.

Definition at line 156 of file limits.

3.11.4 Function Documentation

3.11.4.1 template<typename _RandomAccessIterator > void std::__final_insertion_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*)

This is a helper function for the sort routine.

Definition at line 2230 of file stl_algo.h.

References __insertion_sort(), and __unguarded_insertion_sort().

Referenced by sort().

3.11.4.2 template<typename _RandomAccessIterator , typename _Compare > void std::__final_insertion_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*)

This is a helper function for the sort routine.

Definition at line 2245 of file stl_algo.h.

References __insertion_sort(), and __unguarded_insertion_sort().

3.11.4.3 template<typename _InputIterator , typename _Tp > _InputIterator std::__find (_InputIterator *__first,* _InputIterator *__last,* const _Tp & *__val,* input_iterator_tag) `[inline]`

This is an overload used by find() for the Input Iterator case.

Definition at line 130 of file stl_algo.h.

Referenced by find().

3.11.4.4 template<typename _RandomAccessIterator , typename _Tp > _RandomAccessIterator std::__find (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* const _Tp & *__val,* random_access_iterator_tag)

This is an overload used by find() for the RAI case.

Definition at line 152 of file stl_algo.h.

3.11.4.5 template<typename _InputIterator , typename _Predicate > _InputIterator std::__find_if (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred,* input_iterator_tag) `[inline]`

This is an overload used by find_if() for the Input Iterator case.

Definition at line 141 of file stl_algo.h.

Referenced by find_if().

3.11.4.6 template<typename _RandomAccessIterator , typename _Predicate > _RandomAccessIterator std::__find_if (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Predicate *__pred,* random_access_iterator_tag)

This is an overload used by find_if() for the RAI case.

Definition at line 200 of file stl_algo.h.

3.11.4.7 template<typename _InputIterator , typename _Predicate > _InputIterator std::__find_if_not (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred,* input_iterator_tag) [inline]

This is an overload used by find_if_not() for the Input Iterator case.

Definition at line 248 of file stl_algo.h.

Referenced by __find_if_not(), find_if_not(), and stable_partition().

3.11.4.8 template<typename _RandomAccessIterator , typename _Predicate > _RandomAccessIterator std::__find_if_not (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Predicate *__pred,* random_access_iterator_tag)

This is an overload used by find_if_not() for the RAI case.

Definition at line 259 of file stl_algo.h.

3.11.4.9 template<typename _InputIterator , typename _Predicate > _InputIterator std::__find_if_not (_InputIterator *__first,* _InputIterator *__last,* _Predicate *__pred*) [inline]

Provided for stable_partition to use.

Definition at line 307 of file stl_algo.h.

References __find_if_not(), and __iterator_category().

3.11.4.10 template<typename _InputIterator , typename _Predicate , typename _Distance > _InputIterator std::__find_if_not_n (_InputIterator *__first,* _Distance & *__len,* _Predicate *__pred*)

Like find_if_not(), but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Definition at line 319 of file stl_algo.h.

Referenced by __inplace_stable_partition(), and __stable_partition_adaptive().

3.11.4.11 template<typename _EuclideanRingElement > _EuclideanRingElement std::__gcd (_EuclideanRingElement *__m,* _EuclideanRingElement *__n*)

This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

Definition at line 1518 of file stl_algo.h.

3.11.4.12 template<typename _RandomAccessIterator > void std::__heap_select (_RandomAccessIterator *__first,* _RandomAccessIterator *__middle,* _RandomAccessIterator *__last*)

This is a helper function for the sort routines.

Definition at line 1953 of file stl_algo.h.

References make_heap().

Referenced by partial_sort().

3.11.4.13 template<typename _RandomAccessIterator , typename _Compare > void std::__heap_select (_RandomAccessIterator *__first,* _RandomAccessIterator *__middle,* _RandomAccessIterator *__last,* _Compare *__comp*)

This is a helper function for the sort routines.

Definition at line 1966 of file stl_algo.h.

References make_heap().

3.11.4.14 template<typename _ForwardIterator , typename _Predicate , typename _Distance > _ForwardIterator std::__inplace_stable_partition (_ForwardIterator *__first,* _Predicate *__pred,* _Distance *__len*)

This is a helper function... Requires __len != 0 and !__pred($*$__first), same as __stable_partition_adaptive.

Definition at line 1809 of file stl_algo.h.

References __find_if_not_n(), advance(), distance(), and rotate().

Referenced by stable_partition().

3.11.4.15 template<typename _RandomAccessIterator > void std::__inplace_stable_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*)

This is a helper function for the stable sorting routines.

Definition at line 3509 of file stl_algo.h.

References __insertion_sort(), and __merge_without_buffer().

Referenced by __inplace_stable_sort(), and stable_sort().

3.11.4.16 template<typename _RandomAccessIterator , typename _Compare > void std::__inplace_stable_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*)

This is a helper function for the stable sorting routines.

Definition at line 3528 of file stl_algo.h.

References __inplace_stable_sort(), __insertion_sort(), and __merge_without_buffer().

3.11.4.17 template<typename _RandomAccessIterator > void std::__insertion_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last*)

This is a helper function for the sort routine.

Definition at line 2153 of file stl_algo.h.

References __unguarded_linear_insert().

Referenced by __final_insertion_sort(), and __inplace_stable_sort().

3.11.4.18 template<typename _RandomAccessIterator , typename _Compare > void std::__insertion_sort (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Compare *__comp*)

This is a helper function for the sort routine.

Definition at line 2176 of file stl_algo.h.

References __unguarded_linear_insert().

3.11.4.19 template<typename _RandomAccessIterator , typename _Size > void std::__introsort_loop (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Size *__depth_limit*)

This is a helper function for the sort routine.

Definition at line 2325 of file stl_algo.h.

References __unguarded_partition_pivot().

Referenced by __introsort_loop(), and sort().

3.11.4.20 template<typename _RandomAccessIterator , typename _Size , typename _Compare > void std::__introsort_loop (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* _Size *__depth_limit,* _Compare *__comp*)

This is a helper function for the sort routine.

Definition at line 2347 of file stl_algo.h.

References __introsort_loop(), and __unguarded_partition_pivot().

3.11.4.21 template<typename _Functor , typename... _Args> enable_if< (is_member_pointer< _Functor >::value &&!is_function< _Functor >::value &&!is_function< typename remove_pointer< _Functor >::type >::value), typename result_of< _Functor(_Args &&...)>::type >::type std::__invoke (_Functor & *__f,* _Args &&... *__args*) `[inline]`

Invoke a function object, which may be either a member pointer or a function object. The first parameter will tell which.

Definition at line 232 of file functional.

3.11.4.22 constexpr int std::__lg (int *__n*) `[inline]`

This is a helper function for the sort routines and for random.tcc.

Definition at line 980 of file stl_algobase.h.

Referenced by nth_element(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed(), and sort().

3.11.4.23 template<typename _BidirectionalIterator , typename _Distance , typename _Pointer > void std::__merge_adaptive (_BidirectionalIterator *__first,* _BidirectionalIterator *__middle,* _BidirectionalIterator *__last,* _Distance *__len1,* _Distance *__len2,* _Pointer *__buffer,* _Distance *__buffer_size*)

This is a helper function for the merge routines.

Definition at line 2957 of file stl_algo.h.

References __move_merge_adaptive(), __move_merge_adaptive_backward(), __rotate_adaptive(), advance(), distance(), lower_bound(), and upper_bound().

Referenced by __merge_adaptive(), and inplace_merge().

3.11.4.24 template<typename _BidirectionalIterator , typename _Distance , typename _Pointer , typename _Compare > void std::__merge_adaptive (_BidirectionalIterator *__first,* _BidirectionalIterator *__middle,* _BidirectionalIterator *__last,* _Distance *__len1,* _Distance *__len2,* _Pointer *__buffer,* _Distance *__buffer_size,* _Compare *__comp*)

This is a helper function for the merge routines.

Definition at line 3013 of file stl_algo.h.

References __merge_adaptive(), __move_merge_adaptive(), __move_merge_adaptive_backward(), __rotate_adaptive(), advance(), distance(), lower_bound(), and upper_bound().

3.11.4.25 template<typename _BidirectionalIterator , typename _Distance > void std::__merge_without_buffer (_BidirectionalIterator *__first*, _BidirectionalIterator *__middle*, _BidirectionalIterator *__last*, _Distance *__len1*, _Distance *__len2*)

This is a helper function for the merge routines.

Definition at line 3070 of file stl_algo.h.

References advance(), distance(), iter_swap(), lower_bound(), rotate(), and upper_bound().

Referenced by __inplace_stable_sort(), __merge_without_buffer(), and inplace_merge().

3.11.4.26 template<typename _BidirectionalIterator , typename _Distance , typename _Compare > void std::__merge_without_buffer (_BidirectionalIterator *__first*, _BidirectionalIterator *__middle*, _BidirectionalIterator *__last*, _Distance *__len1*, _Distance *__len2*, _Compare *__comp*)

This is a helper function for the merge routines.

Definition at line 3114 of file stl_algo.h.

References __merge_without_buffer(), advance(), distance(), iter_swap(), lower_bound(), rotate(), and upper_bound().

3.11.4.27 template<typename _Iterator > void std::__move_median_first (_Iterator *__a*, _Iterator *__b*, _Iterator *__c*)

Swaps the median value of $*$__a, $*$__b and $*$__c to $*$__a.

Definition at line 78 of file stl_algo.h.

References iter_swap().

Referenced by __unguarded_partition_pivot().

3.11.4.28 template<typename _Iterator , typename _Compare > void std::__move_median_first (_Iterator *__a*, _Iterator *__b*, _Iterator *__c*, _Compare *__comp*)

Swaps the median value of $*$__a, $*$__b and $*$__c under __comp to $*$__a.

Definition at line 102 of file stl_algo.h.

References iter_swap().

3.11.4.29 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > _OutputIterator std::__move_merge (_InputIterator1 *__first1*, _InputIterator1 *__last1*, _InputIterator2 *__first2*, _InputIterator2 *__last2*, _OutputIterator *__result*)

This is a helper function for the __merge_sort_loop routines.

Definition at line 3272 of file stl_algo.h.

3.11.4.30 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > _OutputIterator std::__move_merge (_InputIterator1 *__first1*, _InputIterator1 *__last1*, _InputIterator2 *__first2*, _InputIterator2 *__last2*, _OutputIterator *__result*, _Compare *__comp*)

This is a helper function for the __merge_sort_loop routines.

Definition at line 3299 of file stl_algo.h.

3.11.4.31 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator > void std::__move_merge_adaptive (_InputIterator1 *__first1*, _InputIterator1 *__last1*, _InputIterator2 *__first2*, _InputIterator2 *__last2*, _OutputIterator *__result*)

This is a helper function for the __merge_adaptive routines.

Definition at line 2778 of file stl_algo.h.

Referenced by __merge_adaptive().

3.11.4.32 template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare > void std::__move_merge_adaptive (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _InputIterator2 *__last2,* _OutputIterator *__result,* _Compare *__comp*)

This is a helper function for the __merge_adaptive routines.

Definition at line 2804 of file stl_algo.h.

3.11.4.33 template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 > void std::__move_merge_adaptive_backward (_BidirectionalIterator1 *__first1,* _BidirectionalIterator1 *__last1,* _BidirectionalIterator2 *__first2,* _BidirectionalIterator2 *__last2,* _BidirectionalIterator3 *__result*)

This is a helper function for the __merge_adaptive routines.

Definition at line 2830 of file stl_algo.h.

Referenced by __merge_adaptive().

3.11.4.34 template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 , typename _Compare > void std::__move_merge_adaptive_backward (_BidirectionalIterator1 *__first1,* _BidirectionalIterator1 *__last1,* _BidirectionalIterator2 *__first2,* _BidirectionalIterator2 *__last2,* _BidirectionalIterator3 *__result,* _Compare *__comp*)

This is a helper function for the __merge_adaptive routines.

Definition at line 2872 of file stl_algo.h.

3.11.4.35 template<typename _ForwardIterator , typename _Predicate > _ForwardIterator std::__partition (_ForwardIterator *__first,* _ForwardIterator *__last,* _Predicate *__pred,* forward_iterator_tag)

This is a helper function...

Definition at line 1752 of file stl_algo.h.

References iter_swap().

Referenced by partition().

3.11.4.36 template<typename _BidirectionalIterator , typename _Predicate > _BidirectionalIterator std::__partition (_BidirectionalIterator *__first,* _BidirectionalIterator *__last,* _Predicate *__pred,* bidirectional_iterator_tag)

This is a helper function...

Definition at line 1777 of file stl_algo.h.

References iter_swap().

3.11.4.37 template<typename _BidirectionalIterator > void std::__reverse (_BidirectionalIterator *__first,* _BidirectionalIterator *__last,* bidirectional_iterator_tag)

This is an uglified reverse(_BidirectionalIterator, _BidirectionalIterator) overloaded for bidirectional iterators.

Definition at line 1418 of file stl_algo.h.

References iter_swap().

Referenced by __rotate(), and reverse().

3.11.4.38 template<typename _RandomAccessIterator > void std::__reverse (_RandomAccessIterator *__first,* _RandomAccessIterator *__last,* random_access_iterator_tag)

This is an uglified reverse(_BidirectionalIterator, _BidirectionalIterator) overloaded for random access iterators.

Definition at line 1438 of file stl_algo.h.

References iter_swap().

3.11.4.39 template<typename _ForwardIterator > void std::__rotate (_ForwardIterator *__first,* _ForwardIterator *__middle,* _ForwardIterator *__last,* forward_iterator_tag)

This is a helper function for the rotate algorithm.

Definition at line 1532 of file stl_algo.h.

References iter_swap().

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object(), and rotate().

3.11.4.40 template<typename _BidirectionalIterator > void std::__rotate (_BidirectionalIterator *__first,* _BidirectionalIterator *__middle,* _BidirectionalIterator *__last,* bidirectional_iterator_tag)

This is a helper function for the rotate algorithm.

Definition at line 1568 of file stl_algo.h.

References __reverse(), and iter_swap().

3.11.4.41 template<typename _RandomAccessIterator > void std::__rotate (_RandomAccessIterator *__first,* _RandomAccessIterator *__middle,* _RandomAccessIterator *__last,* random_access_iterator_tag)

This is a helper function for the rotate algorithm.

Definition at line 1598 of file stl_algo.h.

References iter_swap(), and swap_ranges().

3.11.4.42 template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance > _BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 *__first,* _BidirectionalIterator1 *__middle,* _BidirectionalIterator1 *__last,* _Distance *__len1,* _Distance *__len2,* _BidirectionalIterator2 *__buffer,* _Distance *__buffer_size*)

This is a helper function for the merge routines.

Definition at line 2915 of file stl_algo.h.

References advance(), distance(), and rotate().

Referenced by __merge_adaptive().

3.11.4.43 template<typename _ForwardIterator , typename _Integer , typename _Tp > _ForwardIterator std::__search_n (_ForwardIterator *__first,* _ForwardIterator *__last,* _Integer *__count,* const _Tp & *__val,* std::forward_iterator_tag)

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&) overloaded for forward iterators.

Definition at line 347 of file stl_algo.h.

Referenced by search_n().

3.11.4.44 template<typename _RandomAccessIter , typename _Integer , typename _Tp > _RandomAccessIter std::__search_n (_RandomAccessIter *__first*, _RandomAccessIter *__last*, _Integer *__count*, const _Tp & *__val*, std::random_access_iterator_tag)

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&) overloaded for random access iterators.

Definition at line 379 of file stl_algo.h.

3.11.4.45 template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate > _ForwardIterator std::__search_n (_ForwardIterator *__first*, _ForwardIterator *__last*, _Integer *__count*, const _Tp & *__val*, _BinaryPredicate *__binary_pred*, std::forward_iterator_tag)

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate) overloaded for forward iterators.

Definition at line 433 of file stl_algo.h.

3.11.4.46 template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _BinaryPredicate > _RandomAccessIter std::__search_n (_RandomAccessIter *__first*, _RandomAccessIter *__last*, _Integer *__count*, const _Tp & *__val*, _BinaryPredicate *__binary_pred*, std::random_access_iterator_tag)

This is an uglified search_n(_ForwardIterator, _ForwardIterator, _Integer, const _Tp&, _BinaryPredicate) overloaded for random access iterators.

Definition at line 472 of file stl_algo.h.

3.11.4.47 template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance > _ForwardIterator std::__stable_partition_adaptive (_ForwardIterator *__first*, _ForwardIterator *__last*, _Predicate *__pred*, _Distance *__len*, _Pointer *__buffer*, _Distance *__buffer_size*)

This is a helper function... Requires __first != __last and !__pred(∗__first) and __len == distance(__first, __last).

!__pred(∗__first) allows us to guarantee that we don't move-assign an element onto itself.

Definition at line 1841 of file stl_algo.h.

References __find_if_not_n(), advance(), distance(), and rotate().

Referenced by stable_partition().

3.11.4.48 template<typename _RandomAccessIterator > void std::__unguarded_insertion_sort (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*) [inline]

This is a helper function for the sort routine.

Definition at line 2198 of file stl_algo.h.

References __unguarded_linear_insert().

Referenced by __final_insertion_sort().

3.11.4.49 template<typename _RandomAccessIterator , typename _Compare > void std::__unguarded_insertion_sort (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*, _Compare *__comp*) [inline]

This is a helper function for the sort routine.

Definition at line 2211 of file stl_algo.h.

References __unguarded_linear_insert().

3.11.4.50 template<typename _RandomAccessIterator > void std::__unguarded_linear_insert (_RandomAccessIterator *__last*)

This is a helper function for the sort routine.

Definition at line 2116 of file stl_algo.h.

Referenced by __insertion_sort(), and __unguarded_insertion_sort().

3.11.4.51 template<typename _RandomAccessIterator , typename _Compare > void std::__unguarded_linear_insert (_RandomAccessIterator *__last*, _Compare *__comp*)

This is a helper function for the sort routine.

Definition at line 2134 of file stl_algo.h.

3.11.4.52 template<typename _RandomAccessIterator , typename _Tp > _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*, const _Tp & *__pivot*)

This is a helper function...

Definition at line 2261 of file stl_algo.h.

References iter_swap().

Referenced by __unguarded_partition_pivot().

3.11.4.53 template<typename _RandomAccessIterator , typename _Tp , typename _Compare > _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*, const _Tp & *__pivot*, _Compare *__comp*)

This is a helper function...

Definition at line 2281 of file stl_algo.h.

References iter_swap().

3.11.4.54 template<typename _RandomAccessIterator > _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*) `[inline]`

This is a helper function...

Definition at line 2302 of file stl_algo.h.

References __move_median_first(), and __unguarded_partition().

Referenced by __introsort_loop().

3.11.4.55 template<typename _RandomAccessIterator , typename _Compare > _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator *__first*, _RandomAccessIterator *__last*, _Compare *__comp*) `[inline]`

This is a helper function...

Definition at line 2314 of file stl_algo.h.

References __move_median_first(), and __unguarded_partition().

3.11.4.56 template<typename _ForwardIterator , typename _OutputIterator > _OutputIterator std::__unique_copy (_ForwardIterator *__first*, _ForwardIterator *__last*, _OutputIterator *__result*, forward_iterator_tag , output_iterator_tag)

This is an uglified unique_copy(_InputIterator, _InputIterator, _OutputIterator) overloaded for forward iterators and output iterator as result.

Definition at line 1270 of file stl_algo.h.

Referenced by unique_copy().

3.11.4.57 template<typename _InputIterator , typename _OutputIterator > _OutputIterator std::__unique_copy (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* input_iterator_tag *,* output_iterator_tag)

This is an uglified unique_copy(_InputIterator, _InputIterator, _OutputIterator) overloaded for input iterators and output iterator as result.

Definition at line 1293 of file stl_algo.h.

3.11.4.58 template<typename _InputIterator , typename _ForwardIterator > _ForwardIterator std::__unique_copy (_InputIterator *__first,* _InputIterator *__last,* _ForwardIterator *__result,* input_iterator_tag *,* forward_iterator_tag)

This is an uglified unique_copy(_InputIterator, _InputIterator, _OutputIterator) overloaded for input iterators and forward iterator as result.

Definition at line 1316 of file stl_algo.h.

3.11.4.59 template<typename _ForwardIterator , typename _OutputIterator , typename _BinaryPredicate > _OutputIterator std::__unique_copy (_ForwardIterator *__first,* _ForwardIterator *__last,* _OutputIterator *__result,* _BinaryPredicate *__binary_pred,* forward_iterator_tag *,* output_iterator_tag)

This is an uglified unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate) overloaded for forward iterators and output iterator as result.

Definition at line 1337 of file stl_algo.h.

3.11.4.60 template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate > _OutputIterator std::__unique_copy (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* _BinaryPredicate *__binary_pred,* input_iterator_tag *,* output_iterator_tag)

This is an uglified unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate) overloaded for input iterators and output iterator as result.

Definition at line 1366 of file stl_algo.h.

3.11.4.61 template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate > _ForwardIterator std::__unique_copy (_InputIterator *__first,* _InputIterator *__last,* _ForwardIterator *__result,* _BinaryPredicate *__binary_pred,* input_iterator_tag *,* forward_iterator_tag)

This is an uglified unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate) overloaded for input iterators and forward iterator as result.

Definition at line 1395 of file stl_algo.h.

3.11.4.62 template<typename _T1 , typename... _Args> void std::_Construct (_T1 ∗ *__p,* _Args &&... *__args*) `[inline]`

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 74 of file stl_construct.h.

3.11.4.63 template<typename _Tp > void std::_Destroy (_Tp ∗ *__pointer*) `[inline]`

Destroy the object pointed to by a pointer type.

Definition at line 92 of file stl_construct.h.

Referenced by std::deque< _Tp, _Alloc >::_M_fill_initialize(), std::deque< _Tp, _Alloc >::_M_range_initialize(), std::vector< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::reserve(), and std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::∼vector().

3.11.4.64 template<typename _ForwardIterator > void std::_Destroy (_ForwardIterator *__first,* _ForwardIterator *__last*) [inline]

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 122 of file stl_construct.h.

3.11.4.65 template<typename _InputIterator , typename _Tp > _Tp std::accumulate (_InputIterator *__first,* _InputIterator *__last,* _Tp *__init*) [inline]

Accumulate values in a range.

Accumulates the values in the range [first,last) using operator+(). The initial value is *init*. The values are processed in order.

Parameters

__first	Start of range.
__last	End of range.
__init	Starting value to add other values to.

Returns

The final sum.

Definition at line 120 of file stl_numeric.h.

Referenced by __gnu_parallel::__parallel_partial_sum_linear().

3.11.4.66 template<typename _InputIterator , typename _Tp , typename _BinaryOperation > _Tp std::accumulate (_InputIterator *__first,* _InputIterator *__last,* _Tp *__init,* _BinaryOperation *__binary_op*) [inline]

Accumulate values in a range with operation.

Accumulates the values in the range [first,last) using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.

Parameters

__first	Start of range.
__last	End of range.
__init	Starting value to add other values to.
__binary_op	Function object to accumulate with.

Returns

The final sum.

Definition at line 146 of file stl_numeric.h.

3.11.4.67 template<typename _Tp > std::complex< _Tp > std::acos (const std::complex< _Tp > & *__z*) [inline]

acos(__z) [8.1.2].

Definition at line 1610 of file complex.

3.11.4.68 template<typename _Tp > std::complex< _Tp > std::acosh (const std::complex< _Tp > & __z) [inline]

acosh(__z) [8.1.5].

Definition at line 1726 of file complex.

3.11.4.69 template<typename _InputIterator , typename _OutputIterator > _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)

Return differences between adjacent values.

Computes the difference between adjacent values in the range [first,last) using operator-() and writes the result to `__result`.

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sums.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 317 of file stl_numeric.h.

3.11.4.70 template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation > _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)

Return differences between adjacent values.

Computes the difference between adjacent values in the range [__first,__last) using the function object `__binary_op` and writes the result to `__result`.

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.
__binary_op	Function object.

Returns

Iterator pointing just beyond the values written to result.

_GLIBCXX_RESOLVE_LIB_DEFECTS DR 539. partial_sum and adjacent_difference should mention requirements

Definition at line 360 of file stl_numeric.h.

3.11.4.71 template<typename _InputIterator , typename _Distance > void std::advance (_InputIterator & __i, _Distance __n) [inline]

A generalization of pointer arithmetic.

Parameters

__i	An input iterator.
__n	The *delta* by which to change `__i`.

Returns

Nothing.

This increments `i` by `n`. For bidirectional and random access iterators, `__n` may be negative, in which case `__i` is decremented.

For random access iterators, this uses their + and - operations and are constant time. For other iterator classes they are linear time.

Definition at line 173 of file stl_iterator_base_funcs.h.

References __iterator_category().

Referenced by __inplace_stable_partition(), __merge_adaptive(), __merge_without_buffer(), __rotate_adaptive(), __stable_partition_adaptive(), std::deque< _Tp, _Alloc >::_M_range_initialize(), equal_range(), __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child(), __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child(), is_permutation(), lower_bound(), partition_point(), and upper_bound().

3.11.4.72 bool std::all ( ) const [noexcept]

Tests whether all the bits are on.

Returns

True if all the bits are set.

Definition at line 1317 of file bitset.

3.11.4.73 bool std::any ( ) const [noexcept]

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

Definition at line 1325 of file bitset.

3.11.4.74 template<typename _Tp > __gnu_cxx::__promote<_Tp>::__type std::arg (_Tp __x) [inline]

Additional overloads [8.1.9].

Definition at line 1824 of file complex.

References arg().

3.11.4.75 template<typename _Tp > std::complex< _Tp > std::asin (const std::complex< _Tp > & __z) [inline]

asin(__z) [8.1.3].

Definition at line 1646 of file complex.

3.11.4.76 template<typename _Tp > std::complex< _Tp > std::asinh (const std::complex< _Tp > & __z) [inline]

asinh(__z) [8.1.6].

Definition at line 1765 of file complex.

3.11.4.77 template<typename _Tp > std::complex< _Tp > std::atan (const std::complex< _Tp > & __z) [inline]

atan(__z) [8.1.4].

Definition at line 1690 of file complex.

3.11.4.78 template<typename _Tp > std::complex< _Tp > std::atanh (const std::complex< _Tp > & __z) [inline]

atanh(__z) [8.1.7].

Definition at line 1809 of file complex.

3.11.4.79 template<class _Container > auto std::begin (_Container & __cont) -> decltype(__cont.begin()) [inline]

Return an iterator pointing to the first element of the container.

Parameters

__cont	Container.

Definition at line 48 of file range_access.h.

3.11.4.80 template<class _Container > auto std::begin (const _Container & __cont) -> decltype(__cont.begin()) [inline]

Return an iterator pointing to the first element of the const container.

Parameters

__cont	Container.

Definition at line 58 of file range_access.h.

3.11.4.81 template<class _Tp , size_t _Nm> _Tp* std::begin (_Tp(&) __arr[_Nm]) [inline]

Return an iterator pointing to the first element of the array.

Parameters

__arr	Array.

Definition at line 87 of file range_access.h.

3.11.4.82 template<class _Tp > constexpr const _Tp* std::begin (initializer_list< _Tp > __ils) [noexcept]

Return an iterator pointing to the first element of the initilizer_list.

Parameters

__ils	Initializer list.

Definition at line 89 of file initializer_list.

Referenced by std::vector< _Tp, _Alloc >::emplace(), std::deque< _Tp, _Alloc >::erase(), __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child(), std::vector< _Tp, _Alloc >::insert(), std::list< _Tp, _Alloc >::merge(), std::vector< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::remove(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::sort(), std::forward_list< _Tp, _Alloc >::unique(), and std::list< _Tp, _Alloc >::unique().

3.11.4.83 **ios_base**& std::boolalpha (ios_base & __*base*) [inline]

Calls base.setf(ios_base::boolalpha).

Definition at line 795 of file ios_base.h.

References __gnu_debug::__base(), and std::ios_base::boolalpha.

3.11.4.84 template<typename _Tp , typename _Tp1 , _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > & __*r*) [inline], [noexcept]

const_pointer_cast

Definition at line 1196 of file shared_ptr_base.h.

3.11.4.85 size_t std::count ( ) const [noexcept]

Returns the number of bits which are set.

Definition at line 1277 of file bitset.

Referenced by is_permutation().

3.11.4.86 template<typename _Tp > **reference_wrapper**<const _Tp> std::cref (const _Tp & __*t*) [inline], [noexcept]

Denotes a const reference should be taken to a variable.

Definition at line 481 of file functional.

Referenced by cref().

3.11.4.87 template<typename _Tp > void std::cref (const _Tp &&) [delete]

Denotes a reference should be taken to a variable.

3.11.4.88 template<typename _Tp > **reference_wrapper**<const _Tp> std::cref (reference_wrapper< _Tp > __*t*) [inline], [noexcept]

Partial specialization.

Definition at line 499 of file functional.

References cref().

3.11.4.89 **ios_base**& std::dec (ios_base & __*base*) [inline]

Calls base.setf(ios_base::dec, ios_base::basefield).

Definition at line 933 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::basefield, std::ios_base::dec, and std::ios_base::setf().

Referenced by operator>>().

3.11.4.90 template<typename _InputIterator > iterator_traits<_InputIterator>::difference_type std::distance (_InputIterator __first, _InputIterator __last) [inline]

A generalization of pointer arithmetic.

Parameters

__first	An input iterator.
__last	An input iterator.

Returns

The distance between them.

Returns `n` such that __first + n == __last. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their + and - operations and are constant time. For other iterator classes they are linear time.

Definition at line 114 of file stl_iterator_base_funcs.h.

References __iterator_category().

Referenced by __inplace_stable_partition(), __merge_adaptive(), __merge_without_buffer(), __rotate_adaptive(), __stable_partition_adaptive(), std::deque< _Tp, _Alloc >::_M_range_initialize(), equal_range(), inplace_merge(), is_heap_until(), is_permutation(), std::sub_match< _Bi_iter >::length(), lower_bound(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children(), partition_point(), std::match_results< _FwdIterT, _Alloc >::position(), __gnu_cxx::random_sample_n(), std::list< __inp, __rebind_inp >::size(), and upper_bound().

3.11.4.91 template<typename _Tp , typename _Tp1 , _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]

dynamic_pointer_cast

Definition at line 1206 of file shared_ptr_base.h.

3.11.4.92 template<class _Container > auto std::end (_Container & __cont) -> decltype(__cont.end()) [inline]

Return an iterator pointing to one past the last element of the container.

Parameters

__cont	Container.

Definition at line 68 of file range_access.h.

3.11.4.93 template<class _Container > auto std::end (const _Container & __cont) -> decltype(__cont.end()) [inline]

Return an iterator pointing to one past the last element of the const container.

Parameters

__cont	Container.

Definition at line 78 of file range_access.h.

3.11.4.94 template<class _Tp , size_t _Nm> _Tp* std::end (_Tp(&) *__arr[_Nm]*) [inline]

Return an iterator pointing to one past the last element of the array.

Parameters

__arr	Array.

Definition at line 97 of file range_access.h.

3.11.4.95 template<class _Tp > constexpr const _Tp* std::end (initializer_list< _Tp > *__ils*) [noexcept]

Return an iterator pointing to one past the last element of the initilizer_list.

Parameters

__ils	Initializer list.

Definition at line 99 of file initializer_list.

Referenced by std::vector< _Tp, _Alloc >::emplace(), std::vector< _Tp, _Alloc >::erase(), std::deque< _Tp, _Alloc >::erase(), std::vector< _Tp, _Alloc >::insert(), std::list< _Tp, _Alloc >::merge(), std::vector< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::remove(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::forward_list< _Tp, _Alloc >::resize(), std::forward_list< _Tp, _Alloc >::unique(), and std::list< _Tp, _Alloc >::unique().

3.11.4.96 template<typename _CharT , typename _Traits > **basic_ostream**<_CharT, _Traits>& std::endl (basic_ostream< _CharT, _Traits > & *__os*) [inline]

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more on this subject.

Definition at line 564 of file ostream.

References flush(), std::basic_ostream< _CharT, _Traits >::put(), and std::basic_ios< _CharT, _Traits >::widen().

3.11.4.97 template<typename _CharT , typename _Traits > **basic_ostream**<_CharT, _Traits>& std::ends (basic_ostream< _CharT, _Traits > & *__os*) [inline]

Write a null character into the output sequence.

Null character is `CharT()` by definition. For CharT of `char`, this correctly writes the ASCII `NUL` character string terminator.

Definition at line 576 of file ostream.

References std::basic_ostream< _CharT, _Traits >::put().

3.11.4.98 template<typename _Tp > _Tp std::fabs (const **std::complex**< _Tp > & *__z*) [inline]

fabs(__z) [8.1.8].

Definition at line 1818 of file complex.

References abs().

3.11.4.99 ios_base& std::fixed (ios_base & __base) [inline]

Calls base.setf(ios_base::fixed, ios_base::floatfield).

Definition at line 958 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::fixed, std::ios_base::floatfield, and std::ios_base::setf().

3.11.4.100 bitset< _Nb >& std::flip ( ) [noexcept]

Toggles every bit to its opposite value.

Definition at line 1104 of file bitset.

3.11.4.101 bitset< _Nb >& std::flip (size_t __position)

Toggles a given bit to its opposite value.

Parameters

__position	The index of the bit.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

Definition at line 1117 of file bitset.

References _Unchecked_flip().

3.11.4.102 template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits>& std::flush (basic_ostream< _CharT, _Traits > & __os) [inline]

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 586 of file ostream.

References std::basic_ostream< _CharT, _Traits >::flush().

Referenced by endl().

3.11.4.103 template<typename _MoneyT > _Get_money<_MoneyT> std::get_money (_MoneyT & __mon, bool __intl = false) [inline]

Extended manipulator for extracting money.

Parameters

__mon	Either long double or a specialization of `basic_string`.
__intl	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *__mon*.

Definition at line 256 of file iomanip.

3.11.4.104 template<typename _Tp > pair<_Tp*, ptrdiff_t> std::get_temporary_buffer (ptrdiff_t __len) [noexcept]

Allocates a temporary buffer.

Parameters

__len	The number of objects of type Tp.

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent Tp objects. The objects themselves are not constructed, of course. A pair<> is returned containing *the buffer s address and capacity (in the units of sizeof(_Tp)), or a pair of 0 values if no storage can be obtained.* Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare len with the .second return value.

Provides the nothrow exception guarantee.

Definition at line 85 of file stl_tempbuf.h.

Referenced by std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer().

3.11.4.105 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & *__is,* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & *__str,* _CharT *__delim*)

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.
__delim	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If *delim* was encountered, it is extracted but not stored into *__str*.

Definition at line 626 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

3.11.4.106 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> basic_istream<_CharT, _Traits>& std::getline (basic_istream< _CharT, _Traits > & *__is,* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) `[inline]`

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from is into *__str* until '

' is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If end of line was encountered, it is extracted but not stored into *__str*.

Definition at line 2567 of file vstring.h.

References getline(), and std::basic_ios< _CharT, _Traits >::widen().

3.11.4.107 template<typename _CharT , typename _Traits , typename _Alloc > basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & *__is*, basic_string< _CharT, _Traits, _Alloc > & *__str*, _CharT *__delim*)

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.
__delim	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If *__delim* was encountered, it is extracted but not stored into *__str*.

Definition at line 1068 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::erase(), std::basic_string< _CharT, _Traits, _Alloc >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by getline().

3.11.4.108 template<typename _CharT , typename _Traits , typename _Alloc > basic_istream<_CharT, _Traits>& std::getline (basic_istream< _CharT, _Traits > & *__is*, basic_string< _CharT, _Traits, _Alloc > & *__str*) `[inline]`

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from is into *__str* until '

' is found, the end of the stream is encountered, or str.max_size() is reached. If __is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If end of line was encountered, it is extracted but not stored into *__str*.

Definition at line 2795 of file basic_string.h.

References getline(), and std::basic_ios< _CharT, _Traits >::widen().

3.11.4.109 ios_base& std::hex (ios_base & *__base*) [inline]

Calls base.setf(ios_base::hex, ios_base::basefield).

Definition at line 941 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::basefield, std::ios_base::hex, and std::ios_base::setf().

Referenced by std::regex_traits< _Ch_type >::value().

3.11.4.110 template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp > _Tp std::inner_product (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _Tp *__init*) [inline]

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using operator+(). The values in the ranges are processed in order.

Parameters

__first1	Start of range 1.
__last1	End of range 1.
__first2	Start of range 2.
__init	Starting value to add other values to.

Returns

The final inner product.

Definition at line 174 of file stl_numeric.h.

3.11.4.111 template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp , typename _BinaryOperation1 , typename _BinaryOperation2 > _Tp std::inner_product (_InputIterator1 *__first1,* _InputIterator1 *__last1,* _InputIterator2 *__first2,* _Tp *__init,* _BinaryOperation1 *__binary_op1,* _BinaryOperation2 *__binary_op2*) [inline]

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

Parameters

__first1	Start of range 1.
__last1	End of range 1.
__first2	Start of range 2.
__init	Starting value to add other values to.
__binary_op1	Function object to accumulate with.
__binary_op2	Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 206 of file stl_numeric.h.

3.11.4.112 ios_base& std::internal (ios_base & __base) [inline]

Calls base.setf(ios_base::internal, ios_base::adjustfield).

Definition at line 908 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::adjustfield, and std::ios_base::internal.

3.11.4.113 template<typename _ForwardIterator , typename _Tp > void std::iota (_ForwardIterator *__first*, _ForwardIterator *__last*, _Tp *__value*)

Create a range of sequentially increasing values.

For each element in the range [first,last) assigns `value` and increments `value` as if by `++value`.

Parameters

__first	Start of range.
__last	End of range.
__value	Starting value.

Returns

Nothing.

Definition at line 82 of file stl_numeric.h.

3.11.4.114 template<typename _CharT > bool std::isalnum (_CharT *__c*, const locale & *__loc*) [inline]

Convenience interface to ctype.is(ctype_base::alnum, __c).

Definition at line 2584 of file locale_facets.h.

3.11.4.115 template<typename _CharT > bool std::isalpha (_CharT *__c*, const locale & *__loc*) [inline]

Convenience interface to ctype.is(ctype_base::alpha, __c).

Definition at line 2560 of file locale_facets.h.

3.11.4.116 template<typename _CharT > bool std::iscntrl (_CharT *__c*, const locale & *__loc*) [inline]

Convenience interface to ctype.is(ctype_base::cntrl, __c).

Definition at line 2542 of file locale_facets.h.

3.11.4.117 template<typename _CharT > bool std::isdigit (_CharT *__c*, const locale & *__loc*) [inline]

Convenience interface to ctype.is(ctype_base::digit, __c).

Definition at line 2566 of file locale_facets.h.

3.11.4.118 template<typename _CharT > bool std::isgraph (_CharT *__c*, const locale & *__loc*) [inline]

Convenience interface to ctype.is(ctype_base::graph, __c).

Definition at line 2590 of file locale_facets.h.

3.11.4.119 template<typename _CharT > bool std::islower (_CharT *__c*, const locale & *__loc*) [inline]

Convenience interface to ctype.is(ctype_base::lower, __c).

Definition at line 2554 of file locale_facets.h.

3.11.4.120 template<typename _CharT > bool std::isprint (_CharT __c, const locale & __loc) [inline]

Convenience interface to ctype.is(ctype_base::print, __c).

Definition at line 2536 of file locale_facets.h.

3.11.4.121 template<typename _CharT > bool std::ispunct (_CharT __c, const locale & __loc) [inline]

Convenience interface to ctype.is(ctype_base::punct, __c).

Definition at line 2572 of file locale_facets.h.

3.11.4.122 template<typename _CharT > bool std::isspace (_CharT __c, const locale & __loc) [inline]

Convenience interface to ctype.is(ctype_base::space, __c).

Definition at line 2530 of file locale_facets.h.

3.11.4.123 template<typename _CharT > bool std::isupper (_CharT __c, const locale & __loc) [inline]

Convenience interface to ctype.is(ctype_base::upper, __c).

Definition at line 2548 of file locale_facets.h.

3.11.4.124 template<typename _CharT > bool std::isxdigit (_CharT __c, const locale & __loc) [inline]

Convenience interface to ctype.is(ctype_base::xdigit, __c).

Definition at line 2578 of file locale_facets.h.

3.11.4.125 ios_base& std::left (ios_base & __base) [inline]

Calls base.setf(ios_base::left, ios_base::adjustfield).

Definition at line 916 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::adjustfield, std::ios_base::left, and std::ios_base::setf().

Referenced by operator<<().

3.11.4.126 ios_base& std::noboolalpha (ios_base & __base) [inline]

Calls base.unsetf(ios_base::boolalpha).

Definition at line 803 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::boolalpha, and std::ios_base::unsetf().

3.11.4.127 bool std::none ( ) const [noexcept]

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

Definition at line 1333 of file bitset.

3.11.4.128 ios_base& std::noshowbase (ios_base & __base) [inline]

Calls base.unsetf(ios_base::showbase).

Definition at line 819 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::showbase, and std::ios_base::unsetf().

3.11.4.129 ios_base& std::noshowpoint (ios_base & *__base*) [inline]

Calls base.unsetf(ios_base::showpoint).

Definition at line 835 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::showpoint, and std::ios_base::unsetf().

3.11.4.130 ios_base& std::noshowpos (ios_base & *__base*) [inline]

Calls base.unsetf(ios_base::showpos).

Definition at line 851 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::showpos, and std::ios_base::unsetf().

3.11.4.131 ios_base& std::noskipws (ios_base & *__base*) [inline]

Calls base.unsetf(ios_base::skipws).

Definition at line 867 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::skipws, and std::ios_base::unsetf().

3.11.4.132 ios_base& std::nounitbuf (ios_base & *__base*) [inline]

Calls base.unsetf(ios_base::unitbuf).

Definition at line 899 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::unitbuf, and std::ios_base::unsetf().

3.11.4.133 ios_base& std::nouppercase (ios_base & *__base*) [inline]

Calls base.unsetf(ios_base::uppercase).

Definition at line 883 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::unsetf(), and std::ios_base::uppercase.

3.11.4.134 ios_base& std::oct (ios_base & *__base*) [inline]

Calls base.setf(ios_base::oct, ios_base::basefield).

Definition at line 949 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::basefield, std::ios_base::oct, and std::ios_base::setf().

Referenced by std::regex_traits< _Ch_type >::value().

3.11.4.135 template<typename _Tp , typename _Seq > bool std::operator!= (const stack< _Tp, _Seq > & *__x*, const stack< _Tp, _Seq > & *__y*) [inline]

Based on operator==.

Definition at line 267 of file stl_stack.h.

3.11.4.136 template<typename _Tp > bool std::operator!= (const _Fwd_list_iterator< _Tp > & *__x*, const _Fwd_list_const_iterator< _Tp > & *__y*) [inline]

Forward list iterator inequality comparison.

Definition at line 267 of file forward_list.h.

3.11.4.137 template<typename _Tp , typename _Seq > bool std::operator!= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]

Based on operator==.

Definition at line 292 of file stl_queue.h.

3.11.4.138 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator!= (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]

Returns !(x == y).

Definition at line 761 of file stl_multiset.h.

3.11.4.139 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator!= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]

Returns !(x == y).

Definition at line 776 of file stl_set.h.

3.11.4.140 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Based on operator==.

Definition at line 887 of file stl_multimap.h.

3.11.4.141 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Based on operator==.

Definition at line 985 of file stl_map.h.

3.11.4.142 bool std::operator!= (const bitset< _Nb > & __rhs) const [noexcept]

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1292 of file bitset.

3.11.4.143 template<typename _Tp , typename _Alloc > bool std::operator!= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]

Based on operator==.

Definition at line 1367 of file forward_list.h.

3.11.4.144 template<typename _Tp , typename _Alloc > bool std::operator!= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]

Based on operator==.

Definition at line 1423 of file stl_vector.h.

3.11.4.145 template<typename _Tp , typename _Alloc > bool std::operator!= (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]

Based on operator==.

Definition at line 1634 of file stl_list.h.

3.11.4.146 template<typename _Tp , typename _Alloc > bool std::operator!= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]

Based on operator==.

Definition at line 1983 of file stl_deque.h.

3.11.4.147 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test difference of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

Definition at line 2532 of file basic_string.h.

3.11.4.148 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test difference of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) != 0. False otherwise.

Definition at line 2544 of file basic_string.h.

3.11.4.149 template<typename _Res , typename... _Args> bool std::operator!= (const function< _Res(_Args...)> & __f, nullptr_t) [inline], [noexcept]

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`false` if the wrapper has no target, `true` otherwise

This function will not throw an exception.

Definition at line 2552 of file functional.

3.11.4.150 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]

Test difference of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

Definition at line 2556 of file basic_string.h.

3.11.4.151 template< typename _Res , typename... _Args > bool std::operator!= (nullptr_t , const function< _Res(_Args...)> & *__f*) [inline], [noexcept]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2558 of file functional.

3.11.4.152 template< size_t _Nb > bitset< _Nb > std::operator& (const bitset< _Nb > & *__x*, const bitset< _Nb > & *__y*) [inline], [noexcept]

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1416 of file bitset.

3.11.4.153 template< typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > & *__lhs*, const basic_string< _CharT, _Traits, _Alloc > & *__rhs*)

Concatenate two strings.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

Definition at line 2365 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::append().

3.11.4.154 template<typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT * *__lhs,* const basic_string< _CharT, _Traits, _Alloc > & *__rhs*)

Concatenate C string and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

Definition at line 692 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

3.11.4.155 template<typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT *__lhs,* const basic_string< _CharT, _Traits, _Alloc > & *__rhs*)

Concatenate character and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

Definition at line 708 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size().

3.11.4.156 template<typename _CharT , typename _Traits , typename _Alloc > basic_string<_CharT, _Traits, _Alloc> std::operator+ (const basic_string< _CharT, _Traits, _Alloc > & *__lhs,* const _CharT * *__rhs*) `[inline]`

Concatenate string and C string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

Definition at line 2402 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::append().

3.11.4.157 template< typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc> std::operator+ (const basic_string< _CharT, _Traits, _Alloc > & *__lhs*, _CharT *__rhs*) [inline]

Concatenate string and character.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

Definition at line 2418 of file basic_string.h.

3.11.4.158 template< typename _Tp , typename _Seq > bool std::operator< (const stack< _Tp, _Seq > & *__x*, const stack< _Tp, _Seq > & *__y*) [inline]

Stack ordering relation.

Parameters

__x	A stack.
__y	A stack of the same type as *x*.

Returns

True iff *x* is lexicographically less than *__y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and std::lexicographical_compare() is usually used to make the determination.

Definition at line 261 of file stl_stack.h.

3.11.4.159 template< typename _Tp , typename _Seq > bool std::operator< (const queue< _Tp, _Seq > & *__x*, const queue< _Tp, _Seq > & *__y*) [inline]

Queue ordering relation.

Parameters

__x	A queue.
__y	A queue of the same type as *x*.

Returns

True iff *__x* is lexicographically less than *__y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and std::lexicographical_compare() is usually used to make the determination.

Definition at line 286 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

3.11.4.160 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator< (const multiset< _Key, _Compare, _Alloc > & *__x*, const multiset< _Key, _Compare, _Alloc > & *__y*) [inline]

Multiset ordering relation.

Parameters

__x	A multiset.
__y	A multiset of the same type as *__x*.

Returns

True iff *__x* is lexicographically less than *__y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 754 of file stl_multiset.h.

3.11.4.161 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator< (const set< _Key, _Compare, _Alloc > & *__x*, const set< _Key, _Compare, _Alloc > & *__y*) [inline]

Set ordering relation.

Parameters

__x	A set.
__y	A set of the same type as *x*.

Returns

True iff *__x* is lexicographically less than *__y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 769 of file stl_set.h.

3.11.4.162 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > & *__x*, const multimap< _Key, _Tp, _Compare, _Alloc > & *__y*) [inline]

Multimap ordering relation.

Parameters

__x	A multimap.
__y	A multimap of the same type as *__x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 880 of file stl_multimap.h.

3.11.4.163 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > & *__x,* const map< _Key, _Tp, _Compare, _Alloc > & *__y*) `[inline]`

Map ordering relation.

Parameters

__x	A map.
__y	A map of the same type as *x*.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 978 of file stl_map.h.

3.11.4.164 template<typename _Tp , typename _Alloc > bool std::operator< (const forward_list< _Tp, _Alloc > & *__lx,* const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Forward list ordering relation.

Parameters

__lx	A forward_list.
__ly	A forward_list of the same type as *__lx*.

Returns

True iff *__lx* is lexicographically less than *__ly*.

This is a total ordering relation. It is linear in the number of elements of the forward lists. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1359 of file forward_list.h.

References lexicographical_compare().

3.11.4.165 template<typename _Tp , typename _Alloc > bool std::operator< (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]

Vector ordering relation.

Parameters

__x	A vector.
__y	A vector of the same type as *__x*.

Returns

True iff *__x* is lexicographically less than *__y*.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1416 of file stl_vector.h.

References std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), and lexicographical_compare().

3.11.4.166 template<typename _Tp , typename _Alloc > bool std::operator< (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]

List ordering relation.

Parameters

__x	A list.
__y	A list of the same type as *__x*.

Returns

True iff *__x* is lexicographically less than *__y*.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with <.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1627 of file stl_list.h.

References lexicographical_compare().

3.11.4.167 template<typename _Tp , typename _Alloc > bool std::operator< (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]

Deque ordering relation.

Parameters

__x	A deque.
__y	A deque of the same type as *__x*.

Returns

True iff *x* is lexicographically less than *__y*.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with $<$.

See std::lexicographical_compare() for how the determination is made.

Definition at line 1975 of file stl_deque.h.

References lexicographical_compare().

3.11.4.168 template< typename _CharT , typename _Traits , typename _Alloc > bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > & *__lhs*, const basic_string< _CharT, _Traits, _Alloc > & *__rhs*) [inline]

Test if string precedes string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2569 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.169 template< typename _CharT , typename _Traits , typename _Alloc > bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > & *__lhs*, const _CharT * *__rhs*) [inline]

Test if string precedes C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2581 of file basic_string.h.

3.11.4.170 template< typename _CharT , typename _Traits , typename _Alloc > bool std::operator< (const _CharT * *__lhs*, const basic_string< _CharT, _Traits, _Alloc > & *__rhs*) [inline]

Test if C string precedes string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

Definition at line 2593 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.171 template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > & *__out,* _CharT *__c*) [inline]

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If __`c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 471 of file ostream.

3.11.4.172 template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > & *__out,* char *__c*) [inline]

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If __`c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 476 of file ostream.

3.11.4.173 template<class _Traits > basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits > & *__out,* char *__c*) [inline]

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 482 of file ostream.

3.11.4.174 template<class _Traits > basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits > & __out, signed char __c) `[inline]`

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 488 of file ostream.

3.11.4.175 template<class _Traits > basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits > & __out, unsigned char __c) `[inline]`

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 493 of file ostream.

3.11.4.176 template<typename _CharT , typename _Traits > basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > & __out, const _CharT * __s) [inline]

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 513 of file ostream.

References std::ios_base::badbit.

3.11.4.177 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > & __out, const char * __s)

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 321 of file ostream.tcc.

References std::ios_base::badbit, and std::basic_ios< _CharT, _Traits >::setstate().

3.11.4.178 template<class _Traits > basic_ostream<char, _Traits>& std::operator<< (basic_ostream< char, _Traits > & __out, const char * __s) [inline]

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 530 of file ostream.

References std::ios_base::badbit.

3.11.4.179 template<class _Traits > **basic_ostream**<char, _Traits>& std::operator<< (basic_ostream< char, _Traits > & *__out,* const signed char ∗ *__s*) `[inline]`

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 543 of file ostream.

3.11.4.180 template<class _Traits > **basic_ostream**<char, _Traits>& std::operator<< (basic_ostream< char, _Traits > & *__out,* const unsigned char ∗ *__s*) `[inline]`

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 548 of file ostream.

3.11.4.181 template<typename _CharT , typename _Traits , typename _Tp > basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > && __os, const _Tp & __x) `[inline]`

Generic inserter for rvalue stream.

Parameters

__os	An input stream.
__x	A reference to the object being inserted.

Returns

os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 602 of file ostream.

3.11.4.182 bitset<_Nb> std::operator<< (size_t __position) const `[noexcept]`

Self-explanatory.

Definition at line 1339 of file bitset.

3.11.4.183 template<class _CharT , class _Traits , size_t _Nb> std::basic_ostream<_CharT, _Traits>& std::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb > & __x)

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1521 of file bitset.

References std::__ctype_abstract_base< _CharT >::widen().

3.11.4.184 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) `[inline]`

Write string to a stream.

Parameters

__os	Output stream.
__str	String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into os following the same rules as for writing a C string.

Definition at line 2521 of file vstring.h.

3.11.4.185 template<typename _CharT , typename _Traits , typename _Alloc > basic_ostream<_CharT, _Traits>& std::operator<< (basic_ostream< _CharT, _Traits > & *__os,* const basic_string< _CharT, _Traits, _Alloc > & *__str*) [inline]

Write string to a stream.

Parameters

__os	Output stream.
__str	String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into os following the same rules as for writing a C string.

Definition at line 2753 of file basic_string.h.

3.11.4.186 bitset<_Nb>& std::operator<<= (size_t *__position*) [noexcept]

Operations on bitsets.

Parameters

__position	The number of places to shift.

These should be self-explanatory.

Definition at line 980 of file bitset.

3.11.4.187 template<typename _Tp , typename _Seq > bool std::operator<= (const stack< _Tp, _Seq > & *__x,* const stack< _Tp, _Seq > & *__y*) [inline]

Based on operator<.

Definition at line 279 of file stl_stack.h.

3.11.4.188 template<typename _Tp , typename _Seq > bool std::operator<= (const queue< _Tp, _Seq > & *__x,* const queue< _Tp, _Seq > & *__y*) [inline]

Based on operator<.

Definition at line 304 of file stl_queue.h.

3.11.4.189 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator<= (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]

Returns !(y < x)

Definition at line 775 of file stl_multiset.h.

3.11.4.190 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator<= (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]

Returns !(y < x)

Definition at line 790 of file stl_set.h.

3.11.4.191 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 901 of file stl_multimap.h.

3.11.4.192 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 999 of file stl_map.h.

3.11.4.193 template<typename _Tp , typename _Alloc > bool std::operator<= (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]

Based on operator<.

Definition at line 1388 of file forward_list.h.

3.11.4.194 template<typename _Tp , typename _Alloc > bool std::operator<= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1435 of file stl_vector.h.

3.11.4.195 template<typename _Tp , typename _Alloc > bool std::operator<= (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1646 of file stl_list.h.

3.11.4.196 template<typename _Tp , typename _Alloc > bool std::operator<= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1997 of file stl_deque.h.

3.11.4.197 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test if string doesn't follow string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

Definition at line 2643 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.198 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > & *__lhs*, const _CharT ∗ *__rhs*) [inline]

Test if string doesn't follow C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

Definition at line 2655 of file basic_string.h.

3.11.4.199 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator<= (const _CharT ∗ *__lhs*, const basic_string< _CharT, _Traits, _Alloc > & *__rhs*) [inline]

Test if C string doesn't follow string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

Definition at line 2667 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.200 template<typename _StateT > bool std::operator== (const fpos< _StateT > & *__lhs*, const fpos< _StateT > & *__rhs*) [inline]

Test if equivalent to another position.

Definition at line 216 of file postypes.h.

3.11.4.201 template<typename _Tp , typename _Seq > bool std::operator== (const stack< _Tp, _Seq > & *__x*, const stack< _Tp, _Seq > & *__y*) [inline]

Stack equality comparison.

Parameters

__x	A stack.
__y	A stack of the same type as *__x*.

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 243 of file stl_stack.h.

3.11.4.202 template<typename _Tp > bool std::operator== (const _Fwd_list_iterator< _Tp > & *__x*, const _Fwd_list_const_iterator< _Tp > & *__y*) [inline]

Forward list iterator equality comparison.

Definition at line 258 of file forward_list.h.

3.11.4.203 template<typename _Tp , typename _Seq > bool std::operator== (const queue< _Tp, _Seq > & *__x*, const queue< _Tp, _Seq > & *__y*) [inline]

Queue equality comparison.

Parameters

__x	A queue.
__y	A queue of the same type as *__x*.

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 268 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

3.11.4.204 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator== (const multiset< _Key, _Compare, _Alloc > & *__x*, const multiset< _Key, _Compare, _Alloc > & *__y*) [inline]

Multiset equality comparison.

Parameters

__x	A multiset.
__y	A multiset of the same type as *__x*.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 737 of file stl_multiset.h.

3.11.4.205 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator== (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]

Set equality comparison.

Parameters

__x	A set.
__y	A set of the same type as *x*.

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 752 of file stl_set.h.

3.11.4.206 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Multimap equality comparison.

Parameters

__x	A multimap.
__y	A multimap of the same type as *__x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 863 of file stl_multimap.h.

3.11.4.207 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Map equality comparison.

Parameters

__x	A map.
__y	A map of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 961 of file stl_map.h.

3.11.4.208 bool std::operator== (const bitset< _Nb > & __*rhs*) const [noexcept]

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1288 of file bitset.

3.11.4.209 template<typename _Tp , typename _Alloc > bool std::operator== (const forward_list< _Tp, _Alloc > & __*lx*, const forward_list< _Tp, _Alloc > & __*ly*)

Forward list equality comparison.

Parameters

__*lx*	A forward_list
__*ly*	A forward_list of the same type as __*lx*.

Returns

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 387 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::cbegin(), and std::forward_list< _Tp, _Alloc >::cend().

3.11.4.210 template<typename _Tp , typename _Alloc > bool std::operator== (const vector< _Tp, _Alloc > & __*x*, const vector< _Tp, _Alloc > & __*y*) [inline]

Vector equality comparison.

Parameters

__*x*	A vector.
__*y*	A vector of the same type as __*x*.

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1399 of file stl_vector.h.

References std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), equal(), and std::vector< _Tp, _Alloc >::size().

3.11.4.211 template< typename _Tp , typename _Alloc > bool std::operator== (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]

List equality comparison.

Parameters

__x	A list.
__y	A list of the same type as *__x*.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1598 of file stl_list.h.

References std::list< _Tp, _Alloc >::begin(), and std::list< _Tp, _Alloc >::end().

3.11.4.212 template< typename _Tp , typename _Alloc > bool std::operator== (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]

Deque equality comparison.

Parameters

__x	A deque.
__y	A deque of the same type as *__x*.

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1957 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin(), std::deque< _Tp, _Alloc >::end(), equal(), and std::deque< _Tp, _Alloc >::size().

3.11.4.213 template< typename _CharT , typename _Traits , typename _Alloc > bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test equivalence of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

Definition at line 2486 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.214 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator== (const _CharT * *__lhs*, const basic_string< _CharT, _Traits, _Alloc > & *__rhs*) [inline]

Test equivalence of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) == 0. False otherwise.

Definition at line 2507 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.215 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > & *__lhs*, const _CharT * *__rhs*) [inline]

Test equivalence of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

Definition at line 2519 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.216 template<typename _Res , typename... _Args> bool std::operator== (const function< _Res(_Args...)> & *__f*, nullptr_t) [inline], [noexcept]

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 2534 of file functional.

3.11.4.217 template<typename _Res , typename... _Args> bool std::operator== (nullptr_t , const function< _Res(_Args...)> & __f) [inline], [noexcept]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2540 of file functional.

3.11.4.218 template<typename _Tp , typename _Seq > bool std::operator> (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y) [inline]

Based on operator<.

Definition at line 273 of file stl_stack.h.

3.11.4.219 template<typename _Tp , typename _Seq > bool std::operator> (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y) [inline]

Based on operator<.

Definition at line 298 of file stl_queue.h.

3.11.4.220 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator> (const multiset< _Key, _Compare, _Alloc > & __x, const multiset< _Key, _Compare, _Alloc > & __y) [inline]

Returns y < x.

Definition at line 768 of file stl_multiset.h.

3.11.4.221 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator> (const set< _Key, _Compare, _Alloc > & __x, const set< _Key, _Compare, _Alloc > & __y) [inline]

Returns y < x.

Definition at line 783 of file stl_set.h.

3.11.4.222 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > & __x, const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 894 of file stl_multimap.h.

3.11.4.223 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > & __x, const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 992 of file stl_map.h.

3.11.4.224 template<typename _Tp , typename _Alloc > bool std::operator> (const forward_list< _Tp, _Alloc > & __lx, const forward_list< _Tp, _Alloc > & __ly) [inline]

Based on operator<.

Definition at line 1374 of file forward_list.h.

3.11.4.225 template<typename _Tp , typename _Alloc > bool std::operator> (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1429 of file stl_vector.h.

3.11.4.226 template<typename _Tp , typename _Alloc > bool std::operator> (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1640 of file stl_list.h.

3.11.4.227 template<typename _Tp , typename _Alloc > bool std::operator> (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1990 of file stl_deque.h.

3.11.4.228 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test if string follows string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

Definition at line 2606 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.229 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]

Test if string follows C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

Definition at line 2618 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.230 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator> (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test if C string follows string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

Definition at line 2630 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.231 template<typename _Tp , typename _Seq > bool std::operator>= (const stack< _Tp, _Seq > & *__x*, const stack< _Tp, _Seq > & *__y*) [inline]

Based on operator<.

Definition at line 285 of file stl_stack.h.

3.11.4.232 template<typename _Tp , typename _Seq > bool std::operator>= (const queue< _Tp, _Seq > & *__x*, const queue< _Tp, _Seq > & *__y*) [inline]

Based on operator<.

Definition at line 310 of file stl_queue.h.

3.11.4.233 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= (const multiset< _Key, _Compare, _Alloc > & *__x*, const multiset< _Key, _Compare, _Alloc > & *__y*) [inline]

Returns !(x < y)

Definition at line 782 of file stl_multiset.h.

3.11.4.234 template<typename _Key , typename _Compare , typename _Alloc > bool std::operator>= (const set< _Key, _Compare, _Alloc > & *__x*, const set< _Key, _Compare, _Alloc > & *__y*) [inline]

Returns !(x < y)

Definition at line 797 of file stl_set.h.

3.11.4.235 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > & *__x*, const multimap< _Key, _Tp, _Compare, _Alloc > & *__y*) [inline]

Based on operator<.

Definition at line 908 of file stl_multimap.h.

3.11.4.236 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > & *__x*, const map< _Key, _Tp, _Compare, _Alloc > & *__y*) [inline]

Based on operator<.

Definition at line 1006 of file stl_map.h.

3.11.4.237 template<typename _Tp , typename _Alloc > bool std::operator>= (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) [inline]

Based on operator<.

Definition at line 1381 of file forward_list.h.

3.11.4.238 template<typename _Tp , typename _Alloc > bool std::operator>= (const vector< _Tp, _Alloc > & __x, const vector< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1441 of file stl_vector.h.

3.11.4.239 template<typename _Tp , typename _Alloc > bool std::operator>= (const list< _Tp, _Alloc > & __x, const list< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 1652 of file stl_list.h.

3.11.4.240 template<typename _Tp , typename _Alloc > bool std::operator>= (const deque< _Tp, _Alloc > & __x, const deque< _Tp, _Alloc > & __y) [inline]

Based on operator<.

Definition at line 2004 of file stl_deque.h.

3.11.4.241 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]

Test if string doesn't precede string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

Definition at line 2680 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.242 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > & __lhs, const _CharT * __rhs) [inline]

Test if string doesn't precede C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

Definition at line 2692 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.243 template<typename _CharT , typename _Traits , typename _Alloc > bool std::operator>= (const _CharT * *__lhs*, const basic_string< _CharT, _Traits, _Alloc > & *__rhs*) [inline]

Test if C string doesn't precede string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

Definition at line 2704 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::compare().

3.11.4.244 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & *__in*, _CharT & *__c*)

Character extractors.

Parameters

__in	An input stream.
__c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *__c*. Otherwise, sets failbit in the input stream.

Definition at line 923 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

3.11.4.245 template<class _Traits > basic_istream<char, _Traits>& std::operator>> (basic_istream< char, _Traits > & *__in*, unsigned char & *__c*) [inline]

Character extractors.

Parameters

__in	An input stream.
__c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *__c*. Otherwise, sets failbit in the input stream.

Definition at line 727 of file istream.

3.11.4.246 template<class _Traits > **basic_istream**<char, _Traits>& std::operator>> (basic_istream< char, _Traits > & *__in,* signed char & *__c*) `[inline]`

Character extractors.

Parameters

__in	An input stream.
__c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *__c*. Otherwise, sets failbit in the input stream.

Definition at line 732 of file istream.

3.11.4.247 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & *__in,* _CharT ∗ *__s*)

Character string extractors.

Parameters

__in	An input stream.
__s	A pointer to a character array.

Returns

__in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at *__s*. `n` is defined as:

- if `width()` is greater than zero, `n` is width() otherwise
- `n` is *the number of elements of the largest array of* ∗
- *`char_type` that can store a terminating `eos`.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 955 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::getloc(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::ios_base::width().

3.11.4.248 template<> **basic_istream**<char>& std::operator>> (basic_istream< char > & *__in,* char * *__s*)

Character string extractors.

Parameters

__in	An input stream.
__s	A pointer to a character array.

Returns

__in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at *__s*. `n` is defined as:

- if `width()` is greater than zero, `n` is width() otherwise
- `n` is *the number of elements of the largest array of* *
- *`char_type` that can store a terminating `eos`.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

3.11.4.249 template<class _Traits > **basic_istream**<char, _Traits>& std::operator>> (basic_istream< char, _Traits > & *__in,* unsigned char * *__s*) `[inline]`

Character string extractors.

Parameters

__in	An input stream.
__s	A pointer to a character array.

Returns

__in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at *__s*. `n` is defined as:

- if `width()` is greater than zero, `n` is width() otherwise
- `n` is *the number of elements of the largest array of* $*$
- *`char_type` that can store a terminating `eos`.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 774 of file istream.

3.11.4.250 template<class _Traits > **basic_istream**<char, _Traits>& std::operator>> (basic_istream< char, _Traits > & *__in*, signed char * *__s*) `[inline]`

Character string extractors.

Parameters

__in	An input stream.
__s	A pointer to a character array.

Returns

__in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at *__s*. `n` is defined as:

- if `width()` is greater than zero, `n` is width() otherwise
- `n` is *the number of elements of the largest array of* $*$
- *`char_type` that can store a terminating `eos`.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()`)

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 779 of file istream.

3.11.4.251 template<typename _CharT , typename _Traits , typename _Tp > basic_istream<_CharT, _Traits>& std::operator>> (basic_istream< _CharT, _Traits > && __is, _Tp & __x) [inline]

Generic extractor for rvalue stream.

Parameters

__is	An input stream.
__x	A reference to the extraction target.

Returns

is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 872 of file istream.

3.11.4.252 bitset<_Nb> std::operator>> (size_t __position) const [noexcept]

Self-explanatory.

Definition at line 1343 of file bitset.

3.11.4.253 template<class _CharT , class _Traits , size_t _Nb> std::basic_istream<_CharT, _Traits>& std::operator>> (std::basic_istream< _CharT, _Traits > & __is, bitset< _Nb > & __x)

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Definition at line 1453 of file bitset.

References std::basic_string< _CharT, _Traits, _Alloc >::empty(), std::basic_string< _CharT, _Traits, _Alloc >::push_back(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::widen().

3.11.4.254 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)

Read stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until whitespace is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased.

Definition at line 551 of file vstring.tcc.

References std::ios_base::getloc(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::ios_base::width().

3.11.4.255 template<typename _CharT , typename _Traits , typename _Alloc > basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & *__is,* basic_string< _CharT, _Traits, _Alloc > & *__str*)

Read stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until whitespace is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased.

Definition at line 996 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::erase(), std::ios_base::getloc(), std::basic_string< _CharT, _Traits, _Alloc >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::ios_base::width().

3.11.4.256 bitset<_Nb>& std::operator>>= (size_t *__position*) [noexcept]

Operations on bitsets.

Parameters

__position	The number of places to shift.

These should be self-explanatory.

Definition at line 993 of file bitset.

3.11.4.257 constexpr bool std::operator[] (size_t *__position*)

Array-indexing support.

Parameters

__position	Index into the bitset.

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

_GLIBCXX_RESOLVE_LIB_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1145 of file bitset.

3.11.4.258 template<size_t _Nb> bitset<_Nb> std::operator^ (const bitset< _Nb > & __x, const bitset< _Nb > & __y) `[inline]`, `[noexcept]`

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1434 of file bitset.

3.11.4.259 template<size_t _Nb> bitset<_Nb> std::operator| (const bitset< _Nb > & __x, const bitset< _Nb > & __y) `[inline]`, `[noexcept]`

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1425 of file bitset.

3.11.4.260 bitset<_Nb> std::operator~ ( ) const `[noexcept]`

See the no-argument flip().

Definition at line 1126 of file bitset.

3.11.4.261 template<typename _InputIterator , typename _OutputIterator > _OutputIterator std::partial_sum (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result*)

Return list of partial sums.

Accumulates the values in the range [first,last) using the + operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.

Returns

Iterator pointing just beyond the values written to __result.

Definition at line 237 of file stl_numeric.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs_pu(), and __gnu_parallel::__sequential_random_shuffle().

3.11.4.262 template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation > _OutputIterator std::partial_sum (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* _BinaryOperation *__binary_op*)

Return list of partial sums.

Accumulates the values in the range [first,last) using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.
__binary_op	Function object.

Returns

Iterator pointing just beyond the values written to __result.

Definition at line 278 of file stl_numeric.h.

3.11.4.263 template<typename _MoneyT > _Put_money<_MoneyT> std::put_money (const _MoneyT & *__mon,* bool *__intl =* `false`) `[inline]`

Extended manipulator for inserting money.

Parameters

__mon	Either long double or a specialization of `basic_string`.
__intl	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *__mon*.

Definition at line 303 of file iomanip.

3.11.4.264 template<typename _Tp > **reference_wrapper**<_Tp> std::ref (_Tp & *__t*) `[inline]`, `[noexcept]`

Denotes a reference should be taken to a variable.

Definition at line 475 of file functional.

Referenced by ref().

3.11.4.265 template<typename _Tp > void std::ref (const _Tp &&) `[delete]`

Denotes a reference should be taken to a variable.

3.11.4.266 template<typename _Tp > **reference_wrapper**<_Tp> std::ref (reference_wrapper< _Tp > *__t*) `[inline]`, `[noexcept]`

Partial specialization.

Definition at line 493 of file functional.

References ref().

3.11.4.267 template<typename _InputIterator , typename _OutputIterator , typename _Tp > _OutputIterator std::replace_copy (_InputIterator *__first,* _InputIterator *__last,* _OutputIterator *__result,* const _Tp & *__old_value,* const _Tp & *__new_value*)

Copy a sequence, replacing each element of one value with another value.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__old_value	The value to be replaced.
__new_value	The replacement value.

Returns

The end of the output sequence, `result+`(last-first).

Copies each element in the input range [__first,__last) to the output range [__result,__result+(__last-__first)) replacing elements equal to `__old_value` with `__new_value`.

Definition at line 3907 of file stl_algo.h.

3.11.4.268 bitset<_Nb>& std::reset ( ) `[noexcept]`

Sets every bit to false.

Definition at line 1079 of file bitset.

3.11.4.269 bitset<_Nb>& std::reset (size_t *__position*)

Sets a given bit to false.

Parameters

__position	The index of the bit.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

Same as writing `set(pos,false)`.

Definition at line 1093 of file bitset.

References _Unchecked_reset().

3.11.4.270 _Resetiosflags std::resetiosflags (ios_base::fmtflags *__mask*) [inline]

Manipulator for `setf`.

Parameters

__mask	A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via *stream.setf(0,__mask)*.

Definition at line 63 of file iomanip.

3.11.4.271 template<typename _Tp > void std::return_temporary_buffer (_Tp * *__p*) [inline]

The companion to get_temporary_buffer().

Parameters

__p	A buffer previously allocated by get_temporary_buffer.

Returns

None.

Frees the memory pointed to by __p.

Definition at line 112 of file stl_tempbuf.h.

Referenced by std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer().

3.11.4.272 ios_base& std::right (ios_base & *__base*) [inline]

Calls base.setf(ios_base::right, ios_base::adjustfield).

Definition at line 924 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::adjustfield, std::ios_base::right, and std::ios_base::setf().

3.11.4.273 ios_base& std::scientific (ios_base & *__base*) [inline]

Calls base.setf(ios_base::scientific, ios_base::floatfield).

Definition at line 966 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::floatfield, std::ios_base::scientific, and std::ios_base::setf().

Referenced by operator<<().

3.11.4.274 bitset<_Nb>& std::set ( ) [noexcept]

Sets every bit to true.

Definition at line 1054 of file bitset.

3.11.4.275 bitset< _Nb >& **std::set** (size_t __*position*, bool __*val* = `true`)

Sets a given bit to a particular value.

Parameters

__position	The index of the bit.
__val	Either true or false, defaults to true.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

Definition at line 1068 of file bitset.

References _Unchecked_set().

3.11.4.276 **new_handler** std::set_new_handler (new_handler) throw ()

Takes a replacement handler as the argument, returns the previous handler.

3.11.4.277 _Setbase std::setbase (int __*base*) `[inline]`

Manipulator for `setf`.

Parameters

__base	A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *__base* is any other value.

Definition at line 124 of file iomanip.

3.11.4.278 template<typename _CharT > _Setfill<_CharT> std::setfill (_CharT __*c*) `[inline]`

Manipulator for `fill`.

Parameters

__c	The new fill character.

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

Definition at line 162 of file iomanip.

3.11.4.279 _Setiosflags std::setiosflags (ios_base::fmtflags __*mask*) `[inline]`

Manipulator for `setf`.

Parameters

__mask	A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *__mask*.

Definition at line 93 of file iomanip.

3.11.4.280 _Setprecision std::setprecision (int __n) `[inline]`

Manipulator for `precision`.

Parameters

__n	The new precision.

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

Definition at line 192 of file iomanip.

3.11.4.281 _Setw std::setw (int __n) `[inline]`

Manipulator for `width`.

Parameters

__n	The new width.

Sent to a stream object, this manipulator calls `width(__n)` for that object.

Definition at line 222 of file iomanip.

3.11.4.282 ios_base& std::showbase (ios_base & __base) `[inline]`

Calls base.setf(ios_base::showbase).

Definition at line 811 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::showbase.

3.11.4.283 ios_base& std::showpoint (ios_base & __base) `[inline]`

Calls base.setf(ios_base::showpoint).

Definition at line 827 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::showpoint.

3.11.4.284 ios_base& std::showpos (ios_base & __base) `[inline]`

Calls base.setf(ios_base::showpos).

Definition at line 843 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::showpos.

3.11.4.285 constexpr size_t std::size ( ) const `[noexcept]`

Returns the total number of bits.

Definition at line 1282 of file bitset.

Referenced by std::deque< _Tp, _Alloc >::_M_new_elements_at_back(), std::deque< _Tp, _Alloc >::_M_new_elements_at_front(), std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::assign(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare(), std::basic_string< _CharT, _Traits, _Alloc >::compare(), std::deque< _Tp, _Alloc >::erase(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find(), std::basic_string< _CharT, _Traits, _Alloc >::find(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_first_of(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_of(), std::vector< _Tp, _Alloc >::operator=(), std::deque< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::reserve(), std::basic_string< _CharT, _Traits, _Alloc >::reserve(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize(), std::basic_string< _CharT, _Traits, _Alloc >::resize(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind(), and std::basic_string< _CharT, _Traits, _Alloc >::rfind().

3.11.4.286 ios_base& std::skipws (ios_base & *__base*) [inline]

Calls base.setf(ios_base::skipws).

Definition at line 859 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::skipws.

Referenced by operator>>().

3.11.4.287 template<typename _Tp , typename _Tp1 , _Lock_policy _Lp> __shared_ptr<_Tp, _Lp> std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > & *__r*) [inline], [noexcept]

static_pointer_cast

Definition at line 1186 of file shared_ptr_base.h.

3.11.4.288 template<typename _Key , typename _Compare , typename _Alloc > void std::swap (multiset< _Key, _Compare, _Alloc > & *__x,* multiset< _Key, _Compare, _Alloc > & *__y*) [inline]

See std::multiset::swap().

Definition at line 789 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::swap().

3.11.4.289 template<typename _Key , typename _Compare , typename _Alloc > void std::swap (set< _Key, _Compare, _Alloc > & *__x,* set< _Key, _Compare, _Alloc > & *__y*) [inline]

See std::set::swap().

Definition at line 804 of file stl_set.h.

References std::set< _Key, _Compare, _Alloc >::swap().

3.11.4.290 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > & *__x,* multimap< _Key, _Tp, _Compare, _Alloc > & *__y*) [inline]

See std::multimap::swap().

Definition at line 915 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::swap().

3.11.4.291 template<typename _Key , typename _Tp , typename _Compare , typename _Alloc > void std::swap (map< _Key, _Tp, _Compare, _Alloc > & *__x,* map< _Key, _Tp, _Compare, _Alloc > & *__y*) [inline]

See std::map::swap().

Definition at line 1013 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::swap().

3.11.4.292 template<typename _Tp , typename _Alloc > void std::swap (forward_list< _Tp, _Alloc > & *__lx,* forward_list< _Tp, _Alloc > & *__ly*) [inline]

See std::forward_list::swap().

Definition at line 1395 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::swap().

3.11.4.293 template<typename _Tp , typename _Alloc > void std::swap (vector< _Tp, _Alloc > & *__x,* vector< _Tp, _Alloc > & *__y*) [inline]

See std::vector::swap().

Definition at line 1447 of file stl_vector.h.

References std::vector< _Tp, _Alloc >::swap().

3.11.4.294 template<typename _Tp , typename _Alloc > void std::swap (list< _Tp, _Alloc > & *__x,* list< _Tp, _Alloc > & *__y*) [inline]

See std::list::swap().

Definition at line 1658 of file stl_list.h.

References std::list< _Tp, _Alloc >::swap().

3.11.4.295 template<typename _Tp , typename _Alloc > void std::swap (deque< _Tp, _Alloc > & *__x,* deque< _Tp, _Alloc > & *__y*) [inline]

See std::deque::swap().

Definition at line 2011 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::swap().

3.11.4.296 template<typename _Res , typename... _Args> void std::swap (function< _Res(_Args...)> & *__x,* function< _Res(_Args...)> & *__y*) [inline]

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 2570 of file functional.

3.11.4.297 template<typename _CharT , typename _Traits , typename _Alloc > void std::swap (basic_string< _CharT, _Traits, _Alloc > & *__lhs,* basic_string< _CharT, _Traits, _Alloc > & *__rhs*) [inline]

Swap contents of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Exchanges the contents of *__lhs* and *__rhs* in constant time.

Definition at line 2717 of file basic_string.h.

References std::basic_string< _CharT, _Traits, _Alloc >::swap().

3.11.4.298 bool std::test (size_t __position) const

Tests the value of a bit.

Parameters

__position	The index of a bit.

Returns

The value at *pos*.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

Definition at line 1303 of file bitset.

References _Unchecked_test().

3.11.4.299 template<class _CharT , class _Traits , class _Alloc > **std::basic_string**< char, **std::char_traits**< char >, **std::allocator**< char > > std::to_string ( ) const

Returns a character interpretation of the bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1179 of file bitset.

3.11.4.300 unsigned long std::to_ulong ( ) const

Returns a numerical interpretation of the bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

Definition at line 1160 of file bitset.

3.11.4.301 template<typename _CharT > _CharT std::tolower (_CharT *__c,* const locale & *__loc*) [inline]

Convenience interface to ctype.tolower(__c).

Definition at line 2602 of file locale_facets.h.

3.11.4.302 template<typename _CharT > _CharT std::toupper (_CharT *__c,* const locale & *__loc*) [inline]

Convenience interface to ctype.toupper(__c).

Definition at line 2596 of file locale_facets.h.

3.11.4.303 template<typename _InputIterator , typename _ForwardIterator > _ForwardIterator std::uninitialized_copy (_InputIterator *__first,* _InputIterator *__last,* _ForwardIterator *__result*) [inline]

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

__result + (__first - __last)

Like copy(), but does not require an initialized output range.

Definition at line 107 of file stl_uninitialized.h.

Referenced by __gnu_parallel::parallel_sort_mwms_pu().

3.11.4.304 template<typename _InputIterator , typename _Size , typename _ForwardIterator > _ForwardIterator std::uninitialized_copy_n (_InputIterator *__first,* _Size *__n,* _ForwardIterator *__result*) [inline]

Copies the range [first,first+n) into result.

Parameters

__first	An input iterator.
__n	The number of elements to copy.
__result	An output iterator.

Returns

__result + __n

Like copy_n(), but does not require an initialized output range.

Definition at line 647 of file stl_uninitialized.h.

References __iterator_category().

3.11.4.305 template<typename _ForwardIterator , typename _Tp > void std::uninitialized_fill (_ForwardIterator *__first,* _ForwardIterator *__last,* const _Tp & *__x*) [inline]

Copies the value x into the range [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.
__x	The source value.

Returns

Nothing.

Like fill(), but does not require an initialized output range.

Definition at line 164 of file stl_uninitialized.h.

3.11.4.306 template<typename _ForwardIterator , typename _Size , typename _Tp > void std::uninitialized_fill_n (_ForwardIterator *__first,* _Size *__n,* const _Tp & *__x*) [inline]

Copies the value x into the range [first,first+n).

Parameters

__first	An input iterator.
__n	The number of copies to make.
__x	The source value.

Returns

Nothing.

Like fill_n(), but does not require an initialized output range.

Definition at line 218 of file stl_uninitialized.h.

3.11.4.307 ios_base& std::unitbuf (ios_base & *__base*) [inline]

Calls base.setf(ios_base::unitbuf).

Definition at line 891 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::unitbuf.

3.11.4.308 ios_base& std::uppercase (ios_base & *__base*) [inline]

Calls base.setf(ios_base::uppercase).

Definition at line 875 of file ios_base.h.

References __gnu_debug::__base(), std::ios_base::setf(), and std::ios_base::uppercase.

3.11.4.309 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > & *__is*)

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass   mc;

std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a std::basic_istream::sentry inside your definition of operator>>.

Definition at line 1016 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::getloc(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

3.11.5 Variable Documentation

3.11.5.1 ios_base::Init std::__ioinit `[static]`

Linked to standard error (buffered)

Definition at line 74 of file iostream.

3.11.5.2 ostream std::cerr

Linked to standard output.

3.11.5.3 istream std::cin

Linked to standard input.

3.11.5.4 ostream std::clog

Linked to standard error (unbuffered)

3.11.5.5 ostream std::cout

Linked to standard input.

3.11.5.6 wostream std::wcerr

Linked to standard output.

3.11.5.7 wistream std::wcin

Linked to standard error (buffered)

3.11.5.8 wostream std::wclog

Linked to standard error (unbuffered)

3.11.5.9 wostream std::wcout

Linked to standard input.

3.12 std::__debug Namespace Reference

Classes

- class bitset

  Class std::bitset with additional safety/checking/debug instrumentation.
- class deque

  Class std::deque with safety/checking/debug instrumentation.
- class forward_list

  Class std::forward_list with safety/checking/debug instrumentation.

- class list

    Class std::list with safety/checking/debug instrumentation.
- class map

    Class std::map with safety/checking/debug instrumentation.
- class multimap

    Class std::multimap with safety/checking/debug instrumentation.
- class multiset

    Class std::multiset with safety/checking/debug instrumentation.
- class set

    Class std::set with safety/checking/debug instrumentation.
- class unordered_map

    Class std::unordered_map with safety/checking/debug instrumentation.
- class unordered_multimap

    Class std::unordered_multimap with safety/checking/debug instrumentation.
- class unordered_multiset

    Class std::unordered_multiset with safety/checking/debug instrumentation.
- class unordered_set

    Class std::unordered_set with safety/checking/debug instrumentation.
- class vector

    Class std::vector with safety/checking/debug instrumentation.

Functions

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator!=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator!=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator!=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator!=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<size_t _Nb>
  bitset< _Nb > **operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator<** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator<** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator<** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator<** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_ostream< _CharT,
  _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator<=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator<=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator<=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator<=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator<=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  bitset< _Nb > **operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > **operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)

- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

3.12.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior. Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The __glibcxx_requires_xxx macros are merely wrappers around the __glibcxx_check_xxx wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

3.12.2 Function Documentation

3.12.2.1 template<typename _Tp , typename _Alloc > bool std::__debug::operator<= (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Based on operator<.

Definition at line 768 of file debug/forward_list.

3.12.2.2 template<typename _Tp , typename _Alloc > bool std::__debug::operator> (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Based on operator<.

Definition at line 754 of file debug/forward_list.

3.12.2.3 template<typename _Tp , typename _Alloc > bool std::__debug::operator>= (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Based on operator<.

Definition at line 761 of file debug/forward_list.

3.12.2.4 template<typename _Tp , typename _Alloc > void std::__debug::swap (forward_list< _Tp, _Alloc > & *__lx*, forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

See std::forward_list::swap().

Definition at line 775 of file debug/forward_list.

3.13 std::__detail Namespace Reference

Classes

- class _Automaton

  Base class for, um, automata. Could be an NFA or a DFA. Your choice.
- struct _Before_begin
- struct _CharMatcher

  Matches a single character.
- class _Compiler

  Builds an NFA from an input iterator interval.
- struct _Default_ranged_hash

Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that h(k, N) = h2(h1(k), N), but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.

- struct _EndTagger

  End state tag.
- struct _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >

  Specialization.
- struct _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >

  Specialization.
- struct _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

  Specialization.
- struct _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

  Specialization.
- struct _Equality_base
- class _Grep_matcher

  Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.
- struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >

  Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.
- struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >

  Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.
- struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

  Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.
- struct _Hash_node< _Value, false >
- struct _Hash_node< _Value, true >
- struct _Hash_node_base
- struct _Hashtable_base
- struct _Hashtable_ebo_helper< _Nm, _Tp, false >

  Specialization not using EBO.
- struct _Hashtable_ebo_helper< _Nm, _Tp, true >

  Specialization using EBO.
- struct _Hashtable_traits
- struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >

  Specialization.
- struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >

  Specialization.
- struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >

  Specialization.
- struct _Insert_base
- struct _List_node_base

  Common part of a node in the list.
- struct _Local_const_iterator

  local const_iterators

- struct _Local_iterator

  local iterators
- struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

  Specialization.
- struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >

  Specialization.
- struct _Map_base
- struct _Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

  Partial specialization, __unique_keys set to false.
- struct _Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

  Partial specialization, __unique_keys set to true.
- struct _Mod_range_hashing

  Default range hashing function: use division to fold a large number into the range [0, N).
- class _Nfa

  struct _Nfa
- struct _Node_const_iterator

  Node const_iterators, used to iterate through all the hashtable.
- struct _Node_iterator

  Node iterators, used to iterate through all the hashtable.
- struct _Node_iterator_base

  Base class for node iterators.
- struct _PatternCursor

  ABC for pattern matching.
- struct _Prime_rehash_policy

  Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.
- struct _RangeMatcher

  Matches a character range (bracket expression)
- struct _Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >

  Specialization.
- struct _Results

  Provides a generic facade for a templated match_results.
- class _Scanner

  struct _Scanner. Scans an input range for regex tokens.
- struct _Scanner_base

  Base class for scanner.
- class _SpecializedCursor

  Provides a cursor into the specific target string.
- class _SpecializedResults

  A _Results facade specialized for wrapping a templated match_results.
- struct _StartTagger

  Start state tag.
- struct _State

  struct _State
- class _StateSeq

  Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Typedefs

- typedef std::shared_ptr < _Automaton > _AutomatonPtr
- typedef std::function< bool(const _PatternCursor &)> _Matcher
- typedef int _StateIdT
- typedef std::set< _StateIdT > _StateSet
- typedef std::stack< _StateIdT, std::vector< _StateIdT > > _StateStack
- typedef std::function< void(const _PatternCursor &, _Results &)> _Tagger

Enumerations

- enum _Opcode { **_S_opcode_unknown**, **_S_opcode_alternative**, **_S_opcode_subexpr_begin**, **_S_opcode_subexpr_end**, **_S_opcode_match**, **_S_opcode_accept** }

Functions

- template<typename _InIter , typename _TraitsT >
  _AutomatonPtr **__compile** (const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)
- template<typename _FwdIterT >
  _SpecializedCursor< _FwdIterT > **__cursor** (const _FwdIterT &__b, const _FwdIterT __e)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **__distance_fw** (_Iterator __first, _Iterator __last, std::input_iterator_tag)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **__distance_fw** (_Iterator __first, _Iterator __last, std::forward_iterator_tag)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **__distance_fw** (_Iterator __first, _Iterator __last)
- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator **__normalize** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__factor)
- bool _AnyMatcher (const _PatternCursor &)
- template<typename _Value , bool _Cache_hash_code>
  bool **operator!=** (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)
- template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>
  bool **operator!=** (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)
- template<typename _Value , bool _Cache_hash_code>
  bool **operator==** (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)
- template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>
  bool **operator==** (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)

Variables

- static const _StateIdT _S_invalid_state_id

3.13.1 Detailed Description

Implementation details not part of the namespace std interface.

3.14 std::__parallel Namespace Reference

Classes

- struct _CRandNumber

  Functor wrapper for std::rand().

Functions

- template<typename _IIter , typename _Tp , typename _Tag >
  _Tp **__accumulate_switch** (_IIter, _IIter, _Tp, _Tag)
- template<typename _IIter , typename _Tp , typename _IteratorTag >
  _Tp **__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _IteratorTag)
- template<typename _IIter , typename _Tp , typename _BinaryOper , typename _Tag >
  _Tp **__accumulate_switch** (_IIter, _IIter, _Tp, _BinaryOper, _Tag)
- template<typename _IIter , typename _Tp , typename _BinaryOperation , typename _IteratorTag >
  _Tp **__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)
- template<typename _RAIter , typename _Tp , typename _BinaryOper >
  _Tp **__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename __RAIter , typename _Tp , typename _BinaryOperation >
  _Tp **__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _OIter , typename _BinaryOper , typename _Tag1 , typename _Tag2 >
  _OIter **__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >
  _OutputIterator **__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIter , typename _IterTag >
  _FIter **__adjacent_find_switch** (_FIter, _FIter, _IterTag)
- template<typename _FIter , typename _BiPredicate , typename _IterTag >
  _FIter **__adjacent_find_switch** (_FIter, _FIter, _BiPredicate, _IterTag)
- template<typename _RAIter , typename _BiPredicate >
  _RAIter **__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)

- template<typename _RAIter >
  _RAIter **__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _FIterator , typename _IteratorTag >
  _FIterator **__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _FIterator , typename _BinaryPredicate , typename _IteratorTag >
  _FIterator **__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)
- template<typename _RAIter , typename _BinaryPredicate >
  _RAIter **__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)
- template<typename _IIter , typename _Predicate , typename _IterTag >
  iterator_traits< _IIter >
  ::difference_type **__count_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter , typename _Predicate >
  iterator_traits< _RAIter >
  ::difference_type **__count_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _Predicate , typename _IteratorTag >
  iterator_traits< _IIter >
  ::difference_type **__count_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _IIter , typename _Tp , typename _IterTag >
  iterator_traits< _IIter >
  ::difference_type **__count_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter , typename _Tp >
  iterator_traits< _RAIter >
  ::difference_type **__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _Tp , typename _IteratorTag >
  iterator_traits< _IIter >
  ::difference_type **__count_switch** (_IIter __begin, _IIter __end, const _Tp &__value, _IteratorTag)
- template<typename _IIter , typename _FIter , typename _IterTag1 , typename _IterTag2 >
  _IIter **__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)
- template<typename _RAIter , typename _FIter , typename _BiPredicate , typename _IterTag >
  _RAIter **__find_first_of_switch** (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)
- template<typename _IIter , typename _FIter , typename _BiPredicate , typename _IterTag1 , typename _IterTag2 >
  _IIter **__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _IIter , typename _FIterator , typename _IteratorTag1 , typename _IteratorTag2 >
  _IIter **__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter , typename _FIterator , typename _BinaryPredicate , typename _IteratorTag >
  _RAIter **__find_first_of_switch** (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)
- template<typename _IIter , typename _FIterator , typename _BinaryPredicate , typename _IteratorTag1 , typename _IteratorTag2 >
  _IIter **__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _Predicate , typename _IterTag >
  _IIter **__find_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _IIter , typename _Predicate , typename _IteratorTag >
  _IIter **__find_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter , typename _Predicate >
  _RAIter **__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)

- template<typename _IIter , typename _Tp , typename _IteratorTag >
  _IIter **__find_switch** (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)
- template<typename _RAIter , typename _Tp >
  _RAIter **__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter , typename _Tp , typename _IterTag >
  _IIter **__find_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _IIter , typename _Function , typename _IteratorTag >
  _Function **__for_each_switch** (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)
- template<typename _RAIter , typename _Function >
  _Function **__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter , typename _Function , typename _IterTag >
  _Function **__for_each_switch** (_IIter, _IIter, _Function, _IterTag)
- template<typename _OIter , typename _Size , typename _Generator , typename _IterTag >
  _OIter **__generate_n_switch** (_OIter, _Size, _Generator, _IterTag)
- template<typename _OutputIterator , typename _Size , typename _Generator , typename _IteratorTag >
  _OutputIterator **__generate_n_switch** (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)
- template<typename _RAIter , typename _Size , typename _Generator >
  _RAIter **__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIter , typename _Generator , typename _IterTag >
  void **__generate_switch** (_FIter, _FIter, _Generator, _IterTag)
- template<typename _FIterator , typename _Generator , typename _IteratorTag >
  void **__generate_switch** (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)
- template<typename _RAIter , typename _Generator >
  void **__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _Tp , typename BinaryFunction1 , typename BinaryFunction2 >
  _Tp **__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 , typename _Tag1 , typename _Tag2 >
  _Tp **__inner_product_switch** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **__inner_product_switch** (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 , typename _IteratorTag1 , typename _IteratorTag2 >
  _Tp **__inner_product_switch** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >
  bool **__lexicographical_compare_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _Predicate >
  bool **__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IterTag1 , typename _IterTag2 >
  bool **__lexicographical_compare_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _FIter , typename _Compare , typename _IterTag >
  _FIter **__max_element_switch** (_FIter, _FIter, _Compare, _IterTag)

- template<typename _FIterator , typename _Compare , typename _IteratorTag >
  _FIterator **__max_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter , typename _Compare >
  _RAIter **__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare >
  _OutputIterator **__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter , typename _Compare , typename _IterTag >
  _FIter **__min_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _FIterator , typename _Compare , typename _IteratorTag >
  _FIterator **__min_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter , typename _Compare >
  _RAIter **__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _Predicate >
  pair< _RAIter1, _RAIter2 > **__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IterTag1 , typename _IterTag2 >
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _IIter , typename _OIter , typename _BinaryOper , typename _Tag1 , typename _Tag2 >
  _OIter **__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >
  _OutputIterator **__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter , typename _Predicate , typename _IterTag >
  _FIter **__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _FIterator , typename _Predicate , typename _IteratorTag >
  _FIterator **__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter , typename _Predicate >
  _RAIter **__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIter , typename _Predicate , typename _Tp , typename _IterTag >
  void **__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)

- template<typename _FIterator , typename _Predicate , typename _Tp , typename _IteratorTag >
  void **__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter , typename _Predicate , typename _Tp >
  void **__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIter , typename _Tp , typename _IterTag >
  void **__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _FIterator , typename _Tp , typename _IteratorTag >
  void **__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter , typename _Tp >
  void **__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter , typename _Integer , typename _Tp , typename _BiPredicate >
  _RAIter **__search_n_switch** (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)
- template<typename _FIter , typename _Integer , typename _Tp , typename _BiPredicate , typename _IterTag >
  _FIter **__search_n_switch** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)
- template<typename _RAIter , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _RAIter **__search_n_switch** (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)
- template<typename _FIterator , typename _Integer , typename _Tp , typename _BinaryPredicate , typename _IteratorTag >
  _FIterator **__search_n_switch** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)
- template<typename _FIter1 , typename _FIter2 , typename _IterTag1 , typename _IterTag2 >
  _FIter1 **__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _BiPredicate >
  _RAIter1 **__search_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter1 , typename _FIter2 , typename _BiPredicate , typename _IterTag1 , typename _IterTag2 >
  _FIter1 **__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _RAIter1 , typename _RAIter2 >
  _RAIter1 **__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1 , typename _FIterator2 , typename _IteratorTag1 , typename _IteratorTag2 >
  _FIterator1 **__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _BinaryPredicate >
  _RAIter1 **__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1 , typename _FIterator2 , typename _BinaryPredicate , typename _IteratorTag1 , typename _IteratorTag2 >
  _FIterator1 **__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **__set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **__set_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **__set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **__set_intersection_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **__set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **__set_symmetric_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **__set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **__set_union_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter , typename _OIter , typename _UnaryOperation , typename _IterTag1 , typename _IterTag2 >
  _OIter **__transform1_switch** (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
- template<typename _RAIIter , typename _RAOIter , typename _UnaryOperation >
  _RAOIter **__transform1_switch** (_RAIIter, _RAIIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation >
  _RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >
  _RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)

- template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BiOperation >
  _RAIter3 **__transform2_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >
  _OIter **__transform2_switch** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BinaryOperation >
  _RAIter3 **__transform2_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >
  _OutputIterator **__transform2_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)
- template<typename _IIter , typename _OutputIterator , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >
  _OutputIterator **__unique_copy_switch** (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter , typename RandomAccessOutputIterator , typename _Predicate >
  RandomAccessOutputIterator **__unique_copy_switch** (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter , typename _OIter , typename _Predicate , typename _IterTag1 , typename _IterTag2 >
  _OIter **__unique_copy_switch** (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter , typename _RandomAccess_OIter , typename _Predicate >
  _RandomAccess_OIter **__unique_copy_switch** (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter , typename _Tp >
  _Tp **accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter , typename _Tp >
  _Tp **accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  _Tp **accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _Tp , typename _BinaryOper >
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper)
- template<typename _IIter , typename _Tp , typename _BinaryOper >
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp , typename _BinaryOperation >
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp , typename _BinaryOper >
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _Tp , typename _BinaryOperation >
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Tp , typename _BinaryOperation >
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)
- template<typename _IIter , typename _OIter >
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter , typename _OIter >
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)

- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter >
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _FIter >
  _FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter >
  _FIter **adjacent_find** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _BiPredicate >
  _FIter **adjacent_find** (_FIter, _FIter, _BiPredicate)
- template<typename _FIter , typename _BiPredicate >
  _FIter **adjacent_find** (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _BinaryPredicate >
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator , typename _BinaryPredicate >
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)

- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 >
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 >
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter , typename _Tp >
  _IIter **find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  _IIter **find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter , typename _FIter >
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIter , typename _BiPredicate >
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIter , typename _BiPredicate >
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate)
- template<typename _IIter , typename _FIter >
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter)
- template<typename _IIter , typename _FIterator >
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIterator , typename _BinaryPredicate >
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIterator , typename _BinaryPredicate >
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)
- template<typename _IIter , typename _FIterator >
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2)
- template<typename _IIter , typename _Predicate >
  _IIter **find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Predicate >
  _IIter **find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter , typename _Function >
  _Function **for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _Iterator , typename _Function >
  _Function **for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _Iterator , typename _Function >
  _Function **for_each** (_Iterator __begin, _Iterator __end, _Function __f)
- template<typename _IIter , typename _Function >
  _Function **for_each** (_IIter, _IIter, _Function)

- template<typename _FIter , typename _Generator >
  void **generate** (_FIter, _FIter, _Generator)
- template<typename _FIter , typename _Generator >
  void **generate** (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Generator >
  void **generate** (_FIter, _FIter, _Generator, __gnu_parallel::_Parallelism)
- template<typename _FIterator , typename _Generator >
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Generator >
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Generator >
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **generate_n** (_OIter, _Size, _Generator)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **generate_n** (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **generate_n** (_OIter, _Size, _Generator, __gnu_parallel::_Parallelism)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename BinaryFunction1 , typename BinaryFunction2 >
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **inner_product** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1 , typename _IIter2 >
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 >
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)

- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _FIter >
  _FIter **max_element** (_FIter, _FIter)
- template<typename _FIter >
  _FIter **max_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter >
  _FIter **max_element** (_FIter, _FIter, __gnu_parallel::_Parallelism)
- template<typename _FIter , typename _Compare >
  _FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter , typename _Compare >
  _FIter **max_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Compare >
  _FIter **max_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIterator >
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator >
  _FIterator **max_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator , typename _Compare >
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare >
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare >
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)
- template<typename _FIter >
  _FIter **min_element** (_FIter, _FIter)
- template<typename _FIter >
  _FIter **min_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)

- template<typename _FIter >
  _FIter **min_element** (_FIter, _FIter, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter , typename _Compare >
  _FIter **min_element** (_FIter, _FIter, _Compare)
- template<typename _FIter , typename _Compare >
  _FIter **min_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Compare >
  _FIter **min_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIterator >
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator >
  _FIterator **min_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator , typename _Compare >
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _RAIter >
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter , typename _Compare >
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _IIter , typename _OIter >
  _OIter **partial_sum** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)

- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **partial_sum** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter >
  _OIter **partial_sum** (_IIter, _IIter, _OIter __result)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **partial_sum** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _FIter , typename _Predicate >
  _FIter **partition** (_FIter, _FIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Predicate >
  _FIter **partition** (_FIter, _FIter, _Predicate)
- template<typename _FIterator , typename _Predicate >
  _FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Predicate >
  _FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)
- template<typename _RAIter >
  void **random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void **random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void **random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _FIter , typename _Tp >
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _FIter , typename _Tp >
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Tp >
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIterator , typename _Tp >
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Tp >
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Tp >
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)

- template<typename _FIter , typename _Predicate , typename _Tp >
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIterator , typename _Predicate , typename _Tp >
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Predicate , typename _Tp >
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Predicate , typename _Tp >
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, __gnu_parallel::sequential_tag)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BiPredicate >
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIter1 , typename _FIter2 , typename _BiPredicate >
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)
- template<typename _FIterator1 , typename _FIterator2 >
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)
- template<typename _FIterator1 , typename _FIterator2 >
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)
- template<typename _FIterator1 , typename _FIterator2 , typename _BinaryPredicate >
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator1 , typename _FIterator2 , typename _BinaryPredicate >
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)
- template<typename _FIter , typename _Integer , typename _Tp >
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Integer , typename _Tp , typename _BiPredicate >
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Integer , typename _Tp >
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &)
- template<typename _FIter , typename _Integer , typename _Tp , typename _BiPredicate >
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)
- template<typename _FIterator , typename _Integer , typename _Tp >
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Integer , typename _Tp >
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)
- template<typename _FIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)

- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
_OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
_OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
_OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
_OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
_OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
_OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
_OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
_OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
_OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
_OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
_OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
_OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
_OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
_OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
_OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
_OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
_OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
_OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
_OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
_OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)

- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare , typename _Parallelism >
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter >
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter , typename _Compare >
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)

- template<typename _RAIter , typename _Compare >
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare , typename _Parallelism >
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter >
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter , typename _Compare >
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation >
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation >
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation >
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _Predicate >
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)

- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out)
- template<typename _IIter , typename _OutputIterator , typename _Predicate >
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter , typename _OIter >
  _OIter **unique_copy** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter >
  _OIter **unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _Predicate)

3.14.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

3.15 std::__profile Namespace Reference

Classes

- class bitset

  Class std::bitset wrapper with performance instrumentation.
- class deque

  Class std::deque wrapper with performance instrumentation.
- class forward_list

  Class std::forward_list wrapper with performance instrumentation.
- class list

  List wrapper with performance instrumentation.
- class map

  Class std::map wrapper with performance instrumentation.
- class multimap

  Class std::multimap wrapper with performance instrumentation.
- class multiset

  Class std::multiset wrapper with performance instrumentation.
- class set

  Class std::set wrapper with performance instrumentation.
- class unordered_map

  Class std::unordered_map wrapper with performance instrumentation.
- class unordered_multimap

  Class std::unordered_multimap wrapper with performance instrumentation.
- class unordered_multiset

  Unordered_multiset wrapper with performance instrumentation.
- class unordered_set

  Unordered_set wrapper with performance instrumentation.

Functions

- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator!=** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator!=** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Key , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator!=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator!=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator!=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator!=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator!=** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool **operator!=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<size_t _Nb>
  bitset< _Nb > **operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _Iterator , typename _Sequence >
  __iterator_tracker< _Iterator,
  _Sequence > **operator+** (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  __iterator_tracker< _IteratorL,
  _Sequence >::difference_type **operator-** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  __iterator_tracker< _Iterator,
  _Sequence >::difference_type **operator-** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)

- template<typename _Tp , typename _Alloc >
  bool **operator**< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator**< (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator**< (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator**< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator**< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator**< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator**< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator**< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator**< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator**< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_ostream< _CharT,
  _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _Tp , typename _Alloc >
  bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator**<= (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator**<= (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator**<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator**<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator**<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator**<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator**<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator**<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)

- template<typename _Tp , typename _Alloc >
  bool **operator<=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator==** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator==** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Key , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool **operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator>** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator>** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)

- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **operator>=** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **operator>=** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **operator>=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **operator>=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **operator>=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_istream< _CharT,
  _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  bitset< _Nb > **operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > **operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _Tp , typename _Alloc >
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)
- template<typename _Key , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

- template<typename _Key , typename _Compare , typename _Allocator >
  void **swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  void **swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)

3.15.1 Detailed Description

GNU profile code, replaces standard behavior with profile behavior.

3.15.2 Function Documentation

3.15.2.1 template<typename _Tp , typename _Alloc > bool std::__profile::operator<= (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Based on operator<.

Definition at line 166 of file profile/forward_list.

3.15.2.2 template<typename _Tp , typename _Alloc > bool std::__profile::operator> (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Based on operator<.

Definition at line 152 of file profile/forward_list.

3.15.2.3 template<typename _Tp , typename _Alloc > bool std::__profile::operator>= (const forward_list< _Tp, _Alloc > & *__lx*, const forward_list< _Tp, _Alloc > & *__ly*) `[inline]`

Based on operator<.

Definition at line 159 of file profile/forward_list.

3.15.2.4 template<typename _Tp , typename _Alloc > void std::__profile::swap (forward_list< _Tp, _Alloc > & __lx, forward_list< _Tp, _Alloc > & __ly) [inline]

See std::forward_list::swap().

Definition at line 173 of file profile/forward_list.

3.16 std::chrono Namespace Reference

Classes

- struct duration

  duration
- struct duration_values

  duration_values
- struct system_clock

  system_clock
- struct time_point

  time_point
- struct treat_as_floating_point

  treat_as_floating_point

Typedefs

- typedef system_clock **high_resolution_clock**
- typedef duration< int, ratio< 3600 > > hours
- typedef duration< int64_t, micro > microseconds
- typedef duration< int64_t, milli > milliseconds
- typedef duration< int, ratio< 60 > > minutes
- typedef duration< int64_t, nano > nanoseconds
- typedef duration< int64_t > seconds
- typedef system_clock **steady_clock**

Functions

- template<typename _ToDur , typename _Rep , typename _Period >
  constexpr enable_if
  < __is_duration< _ToDur >
  ::value, _ToDur >::type duration_cast (const duration< _Rep, _Period > &__d)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **operator!=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **operator!=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period , typename _Rep2 >
  constexpr duration< typename
  __common_rep_type< _Rep1,
  typename enable_if
  <!__is_duration< _Rep2 >
  ::value, _Rep2 >::type >::type,
  _Period > **operator%** (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type
  < duration< _Rep1, _Period1 >
  , duration< _Rep2, _Period2 >
  >::type **operator%** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1 , typename _Period , typename _Rep2 >
  constexpr duration< typename
  __common_rep_type< _Rep1,
  _Rep2 >::type, _Period > **operator∗** (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1 , typename _Rep2 , typename _Period >
  constexpr duration< typename
  __common_rep_type< _Rep2,
  _Rep1 >::type, _Period > **operator∗** (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type
  < duration< _Rep1, _Period1 >
  , duration< _Rep2, _Period2 >
  >::type **operator+** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Rep2 , typename _Period2 >
  constexpr time_point< _Clock,
  typename common_type< _Dur1,
  duration< _Rep2, _Period2 >
  >::type > **operator+** (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Clock , typename _Dur2 >
  constexpr time_point< _Clock,
  typename common_type< duration
  < _Rep1, _Period1 >, _Dur2 >
  ::type > **operator+** (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type
  < duration< _Rep1, _Period1 >
  , duration< _Rep2, _Period2 >
  >::type **operator-** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Rep2 , typename _Period2 >
  constexpr time_point< _Clock,
  typename common_type< _Dur1,
  duration< _Rep2, _Period2 >
  >::type > **operator-** (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr common_type< _Dur1,
  _Dur2 >::type **operator-** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period , typename _Rep2 >
  constexpr duration< typename
  __common_rep_type< _Rep1,
  typename enable_if
  <!__is_duration< _Rep2 >
  ::value, _Rep2 >::type >::type,
  _Period > **operator/** (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type< _Rep1,
  _Rep2 >::type **operator/** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr bool **operator<** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
constexpr bool **operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr bool **operator<=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
constexpr bool **operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr bool **operator==** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
constexpr bool **operator==** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr bool **operator>** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
constexpr bool **operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr bool **operator>=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
constexpr bool **operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _ToDur , typename _Clock , typename _Dur >
constexpr enable_if
< __is_duration< _ToDur >
::value, time_point< _Clock,
_ToDur > >::type time_point_cast (const time_point< _Clock, _Dur > &__t)

3.16.1 Detailed Description

ISO C++ 2011 entities sub-namespace for time and date.

3.16.2 Typedef Documentation

3.16.2.1 typedef duration<int, ratio<3600> > std::chrono::hours

hours

Definition at line 541 of file chrono.

3.16.2.2 typedef duration<int64_t, micro> std::chrono::microseconds

microseconds

Definition at line 529 of file chrono.

3.16.2.3 typedef **duration**< int64_t, milli > **std::chrono::milliseconds**

milliseconds

Definition at line 532 of file chrono.

3.16.2.4 typedef **duration**< int, **ratio**< 60 > > **std::chrono::minutes**

minutes

Definition at line 538 of file chrono.

3.16.2.5 typedef **duration**< int64_t, nano > **std::chrono::nanoseconds**

nanoseconds

Definition at line 526 of file chrono.

3.16.2.6 typedef **duration**< int64_t > **std::chrono::seconds**

seconds

Definition at line 535 of file chrono.

3.16.3 Function Documentation

3.16.3.1 template< typename _ToDur , typename _Rep , typename _Period > constexpr **enable_if**< __is_duration< _ToDur >::value, _ToDur >::type std::chrono::duration_cast (const duration< _Rep, _Period > & *__d*)

duration_cast

Definition at line 193 of file chrono.

Referenced by std::this_thread::sleep_for().

3.16.3.2 template< typename _ToDur , typename _Clock , typename _Dur > constexpr **enable_if**< __is_duration< _ToDur >::value, **time_point**< _Clock, _ToDur > >::type std::chrono::time_point_cast (const time_point< _Clock, _Dur > & *__t*)

time_point_cast

Definition at line 602 of file chrono.

3.17 std::decimal Namespace Reference

Classes

- class decimal128

  3.2.4 Class decimal128.
- class decimal32

  3.2.2 Class decimal32.
- class decimal64

  3.2.3 Class decimal64.

Functions

- double **decimal128_to_double** (decimal128 __d)

- float **decimal128_to_float** (decimal128 __d)
- long double **decimal128_to_long_double** (decimal128 __d)
- long long **decimal128_to_long_long** (decimal128 __d)
- double **decimal32_to_double** (decimal32 __d)
- float **decimal32_to_float** (decimal32 __d)
- long double **decimal32_to_long_double** (decimal32 __d)
- long long decimal32_to_long_long (decimal32 __d)
- double **decimal64_to_double** (decimal64 __d)
- float **decimal64_to_float** (decimal64 __d)
- long double **decimal64_to_long_double** (decimal64 __d)
- long long **decimal64_to_long_long** (decimal64 __d)
- double **decimal_to_double** (decimal32 __d)
- double **decimal_to_double** (decimal64 __d)
- double **decimal_to_double** (decimal128 __d)
- float **decimal_to_float** (decimal32 __d)
- float **decimal_to_float** (decimal64 __d)
- float **decimal_to_float** (decimal128 __d)
- long double **decimal_to_long_double** (decimal32 __d)
- long double **decimal_to_long_double** (decimal64 __d)
- long double **decimal_to_long_double** (decimal128 __d)
- long long **decimal_to_long_long** (decimal32 __d)
- long long **decimal_to_long_long** (decimal64 __d)
- long long **decimal_to_long_long** (decimal128 __d)
- static decimal128 **make_decimal128** (long long __coeff, int __exp)
- static decimal128 **make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **make_decimal32** (long long __coeff, int __exp)
- static decimal32 **make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **make_decimal64** (long long __coeff, int __exp)
- static decimal64 **make_decimal64** (unsigned long long __coeff, int __exp)
- bool **operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal32 __lhs, int __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal32 __lhs, long __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal32 __lhs, long long __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator!=** (long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator!=** (long long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal64 __lhs, int __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal64 __lhs, long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long __rhs)

- bool **operator!=** (decimal64 __lhs, long long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator!=** (long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator!=** (long long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal128 __lhs, int __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal128 __lhs, long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal128 __lhs, long long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator!=** (long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator!=** (long long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator*** (decimal32 __lhs, int __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator*** (int __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **operator*** (long __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **operator*** (long long __lhs, decimal32 __rhs)
- decimal32 **operator*** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator*** (decimal64 __lhs, int __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator*** (decimal64 __lhs, long __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator*** (decimal64 __lhs, long long __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator*** (int __lhs, decimal64 __rhs)
- decimal64 **operator*** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **operator*** (long __lhs, decimal64 __rhs)
- decimal64 **operator*** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **operator*** (long long __lhs, decimal64 __rhs)
- decimal64 **operator*** (unsigned long long __lhs, decimal64 __rhs)

- decimal128 **operator∗** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **operator∗** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **operator∗** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator∗** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator∗** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator∗** (decimal128 __lhs, int __rhs)
- decimal128 **operator∗** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator∗** (decimal128 __lhs, long __rhs)
- decimal128 **operator∗** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator∗** (decimal128 __lhs, long long __rhs)
- decimal128 **operator∗** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator∗** (int __lhs, decimal128 __rhs)
- decimal128 **operator∗** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **operator∗** (long __lhs, decimal128 __rhs)
- decimal128 **operator∗** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **operator∗** (long long __lhs, decimal128 __rhs)
- decimal128 **operator∗** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator+** (decimal32 __rhs)
- decimal64 **operator+** (decimal64 __rhs)
- decimal128 **operator+** (decimal128 __rhs)
- decimal32 **operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **operator+** (decimal32 __lhs, int __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator+** (decimal32 __lhs, long __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator+** (int __lhs, decimal32 __rhs)
- decimal32 **operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **operator+** (long __lhs, decimal32 __rhs)
- decimal32 **operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **operator+** (long long __lhs, decimal32 __rhs)
- decimal32 **operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator+** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal64 **operator+** (decimal64 __lhs, int __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator+** (decimal64 __lhs, long __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator+** (int __lhs, decimal64 __rhs)
- decimal64 **operator+** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **operator+** (long __lhs, decimal64 __rhs)
- decimal64 **operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **operator+** (long long __lhs, decimal64 __rhs)
- decimal128 **operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal32 __rhs)

- decimal128 **operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator+** (decimal128 __lhs, int __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator+** (decimal128 __lhs, long __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator+** (decimal128 __lhs, long long __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator+** (int __lhs, decimal128 __rhs)
- decimal128 **operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **operator+** (long __lhs, decimal128 __rhs)
- decimal128 **operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **operator+** (long long __lhs, decimal128 __rhs)
- decimal128 **operator+** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator-** (decimal32 __rhs)
- decimal64 **operator-** (decimal64 __rhs)
- decimal128 **operator-** (decimal128 __rhs)
- decimal32 **operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **operator-** (decimal32 __lhs, int __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator-** (decimal32 __lhs, long __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator-** (int __lhs, decimal32 __rhs)
- decimal32 **operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **operator-** (long __lhs, decimal32 __rhs)
- decimal32 **operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **operator-** (long long __lhs, decimal32 __rhs)
- decimal32 **operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator-** (decimal64 __lhs, int __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator-** (decimal64 __lhs, long __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator-** (int __lhs, decimal64 __rhs)
- decimal64 **operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **operator-** (long __lhs, decimal64 __rhs)
- decimal64 **operator-** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **operator-** (long long __lhs, decimal64 __rhs)
- decimal64 **operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator-** (decimal128 __lhs, int __rhs)

- decimal128 **operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator-** (decimal128 __lhs, long __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator-** (int __lhs, decimal128 __rhs)
- decimal128 **operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **operator-** (long __lhs, decimal128 __rhs)
- decimal128 **operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **operator-** (long long __lhs, decimal128 __rhs)
- decimal128 **operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator/** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **operator/** (decimal32 __lhs, int __rhs)
- decimal32 **operator/** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator/** (decimal32 __lhs, long __rhs)
- decimal32 **operator/** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator/** (decimal32 __lhs, long long __rhs)
- decimal32 **operator/** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator/** (int __lhs, decimal32 __rhs)
- decimal32 **operator/** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **operator/** (long __lhs, decimal32 __rhs)
- decimal32 **operator/** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **operator/** (long long __lhs, decimal32 __rhs)
- decimal32 **operator/** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator/** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **operator/** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator/** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator/** (decimal64 __lhs, int __rhs)
- decimal64 **operator/** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator/** (decimal64 __lhs, long __rhs)
- decimal64 **operator/** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator/** (decimal64 __lhs, long long __rhs)
- decimal64 **operator/** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator/** (int __lhs, decimal64 __rhs)
- decimal64 **operator/** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **operator/** (long __lhs, decimal64 __rhs)
- decimal64 **operator/** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **operator/** (long long __lhs, decimal64 __rhs)
- decimal64 **operator/** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator/** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **operator/** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator/** (decimal128 __lhs, long __rhs)
- decimal128 **operator/** (long long __lhs, decimal128 __rhs)
- decimal128 **operator/** (decimal128 __lhs, int __rhs)
- decimal128 **operator/** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator/** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator/** (decimal128 __lhs, long long __rhs)
- decimal128 **operator/** (decimal128 __lhs, unsigned long long __rhs)

- decimal128 **operator/** (int __lhs, decimal128 __rhs)
- decimal128 **operator/** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **operator/** (long __lhs, decimal128 __rhs)
- decimal128 **operator/** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **operator/** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator<** (unsigned long __lhs, decimal32 __rhs)
- bool **operator<** (decimal32 __lhs, decimal32 __rhs)
- bool **operator<** (decimal32 __lhs, decimal64 __rhs)
- bool **operator<** (decimal32 __lhs, decimal128 __rhs)
- bool **operator<** (decimal32 __lhs, int __rhs)
- bool **operator<** (decimal32 __lhs, long __rhs)
- bool **operator<** (decimal32 __lhs, unsigned long __rhs)
- bool **operator<** (decimal32 __lhs, long long __rhs)
- bool **operator<** (int __lhs, decimal32 __rhs)
- bool **operator<** (long __lhs, decimal32 __rhs)
- bool **operator<** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator<** (long long __lhs, decimal32 __rhs)
- bool **operator<** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator<** (unsigned int __lhs, decimal32 __rhs)
- bool **operator<** (decimal32 __lhs, unsigned int __rhs)
- bool **operator<** (long __lhs, decimal64 __rhs)
- bool **operator<** (unsigned long __lhs, decimal64 __rhs)
- bool **operator<** (decimal64 __lhs, decimal64 __rhs)
- bool **operator<** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator<** (long long __lhs, decimal64 __rhs)
- bool **operator<** (decimal64 __lhs, decimal32 __rhs)
- bool **operator<** (decimal64 __lhs, decimal128 __rhs)
- bool **operator<** (decimal64 __lhs, unsigned int __rhs)
- bool **operator<** (decimal64 __lhs, int __rhs)
- bool **operator<** (int __lhs, decimal64 __rhs)
- bool **operator<** (unsigned int __lhs, decimal64 __rhs)
- bool **operator<** (decimal64 __lhs, long long __rhs)
- bool **operator<** (decimal64 __lhs, long __rhs)
- bool **operator<** (decimal64 __lhs, unsigned long __rhs)
- bool **operator<** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator<** (unsigned long __lhs, decimal128 __rhs)
- bool **operator<** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator<** (decimal128 __lhs, unsigned int __rhs)
- bool **operator<** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator<** (decimal128 __lhs, decimal32 __rhs)
- bool **operator<** (int __lhs, decimal128 __rhs)
- bool **operator<** (unsigned int __lhs, decimal128 __rhs)
- bool **operator<** (long long __lhs, decimal128 __rhs)
- bool **operator<** (long __lhs, decimal128 __rhs)
- bool **operator<** (decimal128 __lhs, unsigned long __rhs)
- bool **operator<** (decimal128 __lhs, int __rhs)
- bool **operator<** (decimal128 __lhs, decimal64 __rhs)
- bool **operator<** (decimal128 __lhs, long __rhs)
- bool **operator<** (decimal128 __lhs, decimal128 __rhs)
- bool **operator<** (decimal128 __lhs, long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long __rhs)

- bool **operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **operator==** (decimal32 __lhs, int __rhs)
- bool **operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **operator==** (decimal32 __lhs, long __rhs)
- bool **operator==** (decimal32 __lhs, long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal32 __rhs)
- bool **operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **operator==** (long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **operator==** (long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator==** (long __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, long long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, int __rhs)
- bool **operator==** (decimal64 __lhs, long __rhs)
- bool **operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **operator==** (int __lhs, decimal128 __rhs)
- bool **operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **operator==** (long __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, long long __rhs)
- bool **operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, int __rhs)
- bool **operator==** (decimal128 __lhs, long __rhs)
- bool **operator>** (unsigned int __lhs, decimal32 __rhs)
- bool **operator>** (long __lhs, decimal32 __rhs)
- bool **operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>** (decimal32 __lhs, decimal32 __rhs)

- bool **operator**> (decimal32 __lhs, decimal64 __rhs)
- bool **operator**> (decimal32 __lhs, long __rhs)
- bool **operator**> (unsigned long __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**> (long long __lhs, decimal32 __rhs)
- bool **operator**> (decimal32 __lhs, unsigned int __rhs)
- bool **operator**> (int __lhs, decimal32 __rhs)
- bool **operator**> (decimal32 __lhs, int __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**> (decimal64 __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**> (long long __lhs, decimal64 __rhs)
- bool **operator**> (int __lhs, decimal64 __rhs)
- bool **operator**> (decimal64 __lhs, unsigned int __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long __rhs)
- bool **operator**> (decimal64 __lhs, decimal128 __rhs)
- bool **operator**> (decimal64 __lhs, long __rhs)
- bool **operator**> (decimal64 __lhs, long long __rhs)
- bool **operator**> (decimal64 __lhs, decimal64 __rhs)
- bool **operator**> (decimal64 __lhs, int __rhs)
- bool **operator**> (long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **operator**> (decimal128 __lhs, decimal128 __rhs)
- bool **operator**> (int __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long __rhs)
- bool **operator**> (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned int __rhs)
- bool **operator**> (unsigned int __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, decimal32 __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**> (decimal128 __lhs, long __rhs)
- bool **operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, int __rhs)
- bool **operator**> (long long __lhs, decimal128 __rhs)
- bool **operator**> (long __lhs, decimal128 __rhs)
- bool **operator**> (decimal128 __lhs, decimal64 __rhs)
- bool **operator**> (decimal128 __lhs, long long __rhs)
- bool **operator**>**=** (long long __lhs, decimal32 __rhs)
- bool **operator**>**=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator**>**=** (decimal32 __lhs, decimal64 __rhs)
- bool **operator**>**=** (decimal32 __lhs, unsigned int __rhs)
- bool **operator**>**=** (decimal32 __lhs, decimal32 __rhs)
- bool **operator**>**=** (decimal32 __lhs, int __rhs)
- bool **operator**>**=** (decimal32 __lhs, decimal128 __rhs)
- bool **operator**>**=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**>**=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator**>**=** (long __lhs, decimal32 __rhs)
- bool **operator**>**=** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**>**=** (decimal32 __lhs, long long __rhs)
- bool **operator**>**=** (int __lhs, decimal32 __rhs)

- bool **operator>=** (decimal32 __lhs, long __rhs)
- bool **operator>=** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator>=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator>=** (decimal64 __lhs, long long __rhs)
- bool **operator>=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator>=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator>=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator>=** (decimal64 __lhs, unsigned long __rhs)
- bool **operator>=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator>=** (long __lhs, decimal64 __rhs)
- bool **operator>=** (decimal64 __lhs, long __rhs)
- bool **operator>=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator>=** (decimal64 __lhs, int __rhs)
- bool **operator>=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator>=** (int __lhs, decimal64 __rhs)
- bool **operator>=** (long long __lhs, decimal64 __rhs)
- bool **operator>=** (decimal128 __lhs, int __rhs)
- bool **operator>=** (int __lhs, decimal128 __rhs)
- bool **operator>=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator>=** (long long __lhs, decimal128 __rhs)
- bool **operator>=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator>=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator>=** (decimal128 __lhs, long __rhs)
- bool **operator>=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator>=** (decimal128 __lhs, long long __rhs)
- bool **operator>=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator>=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator>=** (long __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal128 __rhs)

3.17.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

3.17.2 Function Documentation

3.17.2.1 long long std::decimal::decimal32_to_long_long (decimal32 __*d*)

Non-conforming extension: Conversion to integral type.

3.18 std::placeholders Namespace Reference

Variables

- const _Placeholder< 1 > **_1**
- const _Placeholder< 10 > **_10**

- const _Placeholder< 11 > **_11**
- const _Placeholder< 12 > **_12**
- const _Placeholder< 13 > **_13**
- const _Placeholder< 14 > **_14**
- const _Placeholder< 15 > **_15**
- const _Placeholder< 16 > **_16**
- const _Placeholder< 17 > **_17**
- const _Placeholder< 18 > **_18**
- const _Placeholder< 19 > **_19**
- const _Placeholder< 2 > **_2**
- const _Placeholder< 20 > **_20**
- const _Placeholder< 21 > **_21**
- const _Placeholder< 22 > **_22**
- const _Placeholder< 23 > **_23**
- const _Placeholder< 24 > **_24**
- const _Placeholder< 25 > **_25**
- const _Placeholder< 26 > **_26**
- const _Placeholder< 27 > **_27**
- const _Placeholder< 28 > **_28**
- const _Placeholder< 29 > **_29**
- const _Placeholder< 3 > **_3**
- const _Placeholder< 4 > **_4**
- const _Placeholder< 5 > **_5**
- const _Placeholder< 6 > **_6**
- const _Placeholder< 7 > **_7**
- const _Placeholder< 8 > **_8**
- const _Placeholder< 9 > **_9**

3.18.1 Detailed Description

ISO C++11 entities sub-namespace for functional.

3.19 std::regex_constants Namespace Reference

5.1 Regular Expression Syntax Options

- enum __syntax_option {
  _S_icase, **_S_nosubs**, **_S_optimize**, **_S_collate**,
  _S_ECMAScript, **_S_basic**, **_S_extended**, **_S_awk**,
  _S_grep, **_S_egrep**, **_S_syntax_last** }
- typedef unsigned int syntax_option_type
- constexpr syntax_option_type icase
- constexpr syntax_option_type nosubs
- constexpr syntax_option_type optimize
- constexpr syntax_option_type collate
- constexpr syntax_option_type ECMAScript
- constexpr syntax_option_type basic
- constexpr syntax_option_type extended
- constexpr syntax_option_type awk
- constexpr syntax_option_type grep
- constexpr syntax_option_type egrep

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum __match_flag {
  _S_not_bol, **_S_not_eol**, **_S_not_bow**, **_S_not_eow**,
  _S_any, **_S_not_null**, **_S_continuous**, **_S_prev_avail**,
  _S_sed, **_S_no_copy**, **_S_first_only**, **_S_match_flag_last** }
- typedef std::bitset
  < _S_match_flag_last > match_flag_type
- constexpr match_flag_type match_default
- constexpr match_flag_type match_not_bol
- constexpr match_flag_type match_not_eol
- constexpr match_flag_type match_not_bow
- constexpr match_flag_type match_not_eow
- constexpr match_flag_type match_any
- constexpr match_flag_type match_not_null
- constexpr match_flag_type match_continuous
- constexpr match_flag_type match_prev_avail
- constexpr match_flag_type format_default
- constexpr match_flag_type format_sed
- constexpr match_flag_type format_no_copy
- constexpr match_flag_type format_first_only

5.3 Error Types

- enum error_type {
  _S_error_collate, **_S_error_ctype**, **_S_error_escape**, **_S_error_backref**,
  _S_error_brack, **_S_error_paren**, **_S_error_brace**, **_S_error_badbrace**,
  _S_error_range, **_S_error_space**, **_S_error_badrepeat**, **_S_error_complexity**,
  _S_error_stack, **_S_error_last** }
- constexpr error_type error_collate (_S_error_collate)
- constexpr error_type error_ctype (_S_error_ctype)
- constexpr error_type error_escape (_S_error_escape)
- constexpr error_type error_backref (_S_error_backref)
- constexpr error_type error_brack (_S_error_brack)
- constexpr error_type error_paren (_S_error_paren)
- constexpr error_type error_brace (_S_error_brace)
- constexpr error_type error_badbrace (_S_error_badbrace)
- constexpr error_type error_range (_S_error_range)
- constexpr error_type error_space (_S_error_space)
- constexpr error_type error_badrepeat (_S_error_badrepeat)
- constexpr error_type error_complexity (_S_error_complexity)
- constexpr error_type error_stack (_S_error_stack)

3.19.1 Detailed Description

ISO C++-0x entities sub namespace for regex.

3.19.2 Typedef Documentation

3.19.2.1 typedef std::bitset<_S_match_flag_last> std::regex_constants::match_flag_type

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 198 of file regex_constants.h.

3.19.2.2 typedef unsigned int std::regex_constants::syntax_option_type

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 80 of file regex_constants.h.

3.19.3 Enumeration Type Documentation

3.19.3.1 enum std::regex_constants::__match_flag

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 175 of file regex_constants.h.

3.19.3.2 enum std::regex_constants::__syntax_option

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 54 of file regex_constants.h.

3.19.3.3 enum std::regex_constants::error_type

The expression contained an invalid collating element name.

Definition at line 49 of file regex_error.h.

3.19.4 Function Documentation

3.19.4.1 constexpr error_type std::regex_constants::error_backref (_S_error_backref)

The expression contained an invalid back reference.

3.19.4.2 constexpr error_type std::regex_constants::error_badbrace (_S_error_badbrace)

The expression contained an invalid range in a {} expression.

3.19.4.3 constexpr error_type std::regex_constants::error_badrepeat (_S_error_badrepeat)

One of *?+{ was not preceded by a valid regular expression.

3.19.4.4 constexpr error_type std::regex_constants::error_brace (_S_error_brace)

The expression contained mismatched { and }

3.19.4.5 constexpr error_type std::regex_constants::error_brack (_S_error_brack)

The expression contained mismatched [and].

3.19.4.6 constexpr error_type std::regex_constants::error_collate (_S_error_collate)

The expression contained an invalid collating element name.

3.19.4.7 constexpr error_type std::regex_constants::error_complexity (_S_error_complexity)

The complexity of an attempted match against a regular expression exceeded a pre-set level.

3.19.4.8 constexpr error_type std::regex_constants::error_ctype (_S_error_ctype)

The expression contained an invalid character class name.

3.19.4.9 constexpr error_type std::regex_constants::error_escape (_S_error_escape)

The expression contained an invalid escaped character, or a trailing escape.

3.19.4.10 constexpr error_type std::regex_constants::error_paren (_S_error_paren)

The expression contained mismatched (and).

3.19.4.11 constexpr error_type std::regex_constants::error_range (_S_error_range)

The expression contained an invalid character range, such as [b-a] in most encodings.

3.19.4.12 constexpr error_type std::regex_constants::error_space (_S_error_space)

There was insufficient memory to convert the expression into a finite state machine.

3.19.4.13 constexpr error_type std::regex_constants::error_stack (_S_error_stack)

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

3.19.5 Variable Documentation

3.19.5.1 constexpr syntax_option_type std::regex_constants::awk

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility awk in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that C-style escape sequences are supported. These sequences are: \\, \a, \b, \f, \n, \r, \t , \v, \', ', and \ddd (where ddd is one, two, or three octal digits).

Definition at line 144 of file regex_constants.h.

3.19.5.2 constexpr syntax_option_type std::regex_constants::basic

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 126 of file regex_constants.h.

3.19.5.3 constexpr syntax_option_type std::regex_constants::collate

Specifies that character ranges of the form [a-b] should be locale sensitive.

Definition at line 107 of file regex_constants.h.

3.19.5.4 constexpr syntax_option_type std::regex_constants::ECMAScript

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 117 of file regex_constants.h.

3.19.5.5 constexpr syntax_option_type std::regex_constants::egrep

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep when given the -E option in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that newlines are treated as whitespace.

Definition at line 160 of file regex_constants.h.

3.19.5.6 constexpr syntax_option_type std::regex_constants::extended

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 134 of file regex_constants.h.

3.19.5.7 constexpr match_flag_type std::regex_constants::format_default

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript replace function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 String.prototype.replace. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- $$ The dollar-sign itself ($)
- $& The matched substring.
- $‘ The portion of *string* that precedes the matched substring. This would be match_results::prefix().
- $’ The portion of *string* that follows the matched substring. This would be match_results::suffix().

- $n The nth capture, where n is in [1,9] and $n is not followed by a decimal digit. If n <= match_results::size() and the nth capture is undefined, use the empty string instead. If n > match_results::size(), the result is implementation-defined.
- $nn The nnth capture, where nn is a two-digit decimal number on [01, 99]. If nn <= match_results::size() and the nth capture is undefined, use the empty string instead. If nn > match_results::size(), the result is implementation-defined.

Definition at line 280 of file regex_constants.h.

3.19.5.8 constexpr match_flag_type std::regex_constants::format_first_only

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 301 of file regex_constants.h.

3.19.5.9 constexpr match_flag_type std::regex_constants::format_no_copy

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 295 of file regex_constants.h.

3.19.5.10 constexpr match_flag_type std::regex_constants::format_sed

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX sed utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 288 of file regex_constants.h.

3.19.5.11 constexpr syntax_option_type std::regex_constants::grep

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to syntax_option_type basic, except that newlines are treated as whitespace.

Definition at line 152 of file regex_constants.h.

3.19.5.12 constexpr syntax_option_type std::regex_constants::icase

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 86 of file regex_constants.h.

3.19.5.13 constexpr match_flag_type std::regex_constants::match_any

If more than one match is possible then any match is an acceptable result.

Definition at line 235 of file regex_constants.h.

3.19.5.14 constexpr match_flag_type std::regex_constants::match_continuous

The expression only matches a sub-sequence that begins at first .

Definition at line 245 of file regex_constants.h.

3.19.5.15 constexpr match_flag_type std::regex_constants::match_default

The default matching rules.

Definition at line 203 of file regex_constants.h.

3.19.5.16 constexpr match_flag_type std::regex_constants::match_not_bol

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 210 of file regex_constants.h.

3.19.5.17 constexpr match_flag_type std::regex_constants::match_not_bow

The expression \b is not matched against the sub-sequence [first,first).

Definition at line 223 of file regex_constants.h.

3.19.5.18 constexpr match_flag_type std::regex_constants::match_not_eol

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character ($) in the regular expression shall not match [last, last).

Definition at line 217 of file regex_constants.h.

3.19.5.19 constexpr match_flag_type std::regex_constants::match_not_eow

The expression \b should not be matched against the sub-sequence [last,last).

Definition at line 229 of file regex_constants.h.

3.19.5.20 constexpr match_flag_type std::regex_constants::match_not_null

The expression does not match an empty sequence.

Definition at line 240 of file regex_constants.h.

3.19.5.21 constexpr match_flag_type std::regex_constants::match_prev_avail

–first is a valid iterator position. When this flag is set then the flags match_not_bol and match_not_bow are ignored by the regular expression algorithms 28.11 and iterators 28.12.

Definition at line 252 of file regex_constants.h.

3.19.5.22 constexpr syntax_option_type std::regex_constants::nosubs

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied match_results structure.

Definition at line 93 of file regex_constants.h.

3.19.5.23 constexpr syntax_option_type std::regex_constants::optimize

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 101 of file regex_constants.h.

3.20 std::rel_ops Namespace Reference

Functions

- template<class _Tp >
  bool operator!= (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
  bool operator<= (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
  bool operator> (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
  bool operator>= (const _Tp &__x, const _Tp &__y)

3.20.1 Detailed Description

The generated relational operators are sequestered here.

3.20.2 Function Documentation

3.20.2.1 template<class _Tp > bool std::rel_ops::operator!= (const _Tp & __x, const _Tp & __y) [inline]

Defines != for arbitrary types, in terms of ==.

Parameters

__x	A thing.
__y	Another thing.

Returns

__x != __y

This function uses == to determine its result.

Definition at line 87 of file stl_relops.h.

3.20.2.2 template<class _Tp > bool std::rel_ops::operator<= (const _Tp & __x, const _Tp & __y) [inline]

Defines <= for arbitrary types, in terms of <.

Parameters

__x	A thing.
__y	Another thing.

Returns

__x <= __y

This function uses < to determine its result.

Definition at line 113 of file stl_relops.h.

3.20.2.3 template<class _Tp > bool std::rel_ops::operator> (const _Tp & __x, const _Tp & __y) [inline]

Defines > for arbitrary types, in terms of <.

Parameters

__x	A thing.
__y	Another thing.

Returns

__x > __y

This function uses < to determine its result.

Definition at line 100 of file stl_relops.h.

3.20.2.4 template<class _Tp > bool std::rel_ops::operator>= (const _Tp & *__x*, const _Tp & *__y*) `[inline]`

Defines >= for arbitrary types, in terms of <.

Parameters

__x	A thing.
__y	Another thing.

Returns

__x >= __y

This function uses < to determine its result.

Definition at line 126 of file stl_relops.h.

3.21 std::this_thread Namespace Reference

Functions

- void **__sleep_for** (chrono::seconds, chrono::nanoseconds)
- thread::id get_id () noexcept
- template<typename _Rep , typename _Period >
  void sleep_for (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock , typename _Duration >
  void sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)
- void yield () noexcept

3.21.1 Detailed Description

ISO C++ 2011 entities sub-namespace for thread. 30.3.2 Namespace this_thread.

3.21.2 Function Documentation

3.21.2.1 thread::id std::this_thread::get_id ( ) `[inline]`, `[noexcept]`

get_id

Definition at line 252 of file thread.

3.21.2.2 template<typename _Rep , typename _Period > void std::this_thread::sleep_for (const chrono::duration< _Rep, _Period > & __rtime) [inline]

sleep_for

Definition at line 269 of file thread.

References std::chrono::duration_cast().

Referenced by sleep_until().

3.21.2.3 template<typename _Clock , typename _Duration > void std::this_thread::sleep_until (const chrono::time_point< _Clock, _Duration > & __atime) [inline]

sleep_until

Definition at line 288 of file thread.

References sleep_for().

3.21.2.4 void std::this_thread::yield ( ) [inline], [noexcept]

yield

Definition at line 256 of file thread.

3.22 std::tr1 Namespace Reference

Namespaces

- namespace __detail

Functions

- template<typename _Tp >
  std::complex< _Tp > **__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_asinh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > acosh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > asinh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tp >
  std::complex< _Tp > atanh (const std::complex< _Tp > &__z)

- template<typename _Tpx , typename _Tpy >
  __gnu_cxx::__promote_2< _Tpx,
  _Tpy >::__type beta (_Tpx __x, _Tpy __y)
- float **betaf** (float __x, float __y)
- long double **betal** (long double __x, long double __y)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)
- float **comp_ellint_1f** (float __k)
- long double **comp_ellint_1l** (long double __k)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)
- float **comp_ellint_2f** (float __k)
- long double **comp_ellint_2l** (long double __k)
- template<typename _Tp , typename _Tpn >
  __gnu_cxx::__promote_2< _Tp,
  _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **comp_ellint_3f** (float __k, float __nu)
- long double **comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa , typename _Tpc , typename _Tp >
  __gnu_cxx::__promote_3< _Tpa,
  _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float **conf_hypergf** (float __a, float __c, float __x)
- long double **conf_hypergl** (long double __a, long double __c, long double __x)
- template<typename _Tp >
  std::complex< _Tp > **conj** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< typename
  __gnu_cxx::__promote< _Tp >
  ::__type > **conj** (_Tp __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_if** (float __nu, float __x)
- long double **cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_jf** (float __nu, float __x)
- long double **cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_kf** (float __nu, float __x)
- long double **cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)
- float **cyl_neumannf** (float __nu, float __x)
- long double **cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp , typename _Tpp >
  __gnu_cxx::__promote_2< _Tp,
  _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)

- float **ellint_1f** (float __k, float __phi)
- long double **ellint_1l** (long double __k, long double __phi)
- template<typename _Tp , typename _Tpp >
  __gnu_cxx::__promote_2< _Tp,
  _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)
- float **ellint_2f** (float __k, float __phi)
- long double **ellint_2l** (long double __k, long double __phi)
- template<typename _Tp , typename _Tpn , typename _Tpp >
  __gnu_cxx::__promote_3< _Tp,
  _Tpn, _Tpp >::__type ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **ellint_3f** (float __k, float __nu, float __phi)
- long double **ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)
- float **expintf** (float __x)
- long double **expintl** (long double __x)
- template<typename _Tp >
  std::complex< _Tp > fabs (const std::complex< _Tp > &__z)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)
- float **hermitef** (unsigned int __n, float __x)
- long double **hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
  __gnu_cxx::__promote_4< _Tpa,
  _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **hypergf** (float __a, float __b, float __c, float __x)
- long double **hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)
- float **laguerref** (unsigned int __n, float __x)
- long double **laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __n, _Tp __x)
- float **legendref** (unsigned int __n, float __x)
- long double **legendrel** (unsigned int __n, long double __x)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)

- template<typename _Tp >
  std::complex< _Tp > **pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  std::complex< _Tp > **pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp >
  std::complex< _Tp > **pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- double **pow** (double __x, double __y)
- float **pow** (float __x, float __y)
- long double **pow** (long double __x, long double __y)
- template<typename _Tp , typename _Up >
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **pow** (_Tp __x, _Up __y)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)
- float **riemann_zetaf** (float __x)
- long double **riemann_zetal** (long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)
- float **sph_besself** (unsigned int __n, float __x)
- long double **sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)
- float **sph_neumannf** (unsigned int __n, float __x)
- long double **sph_neumannl** (unsigned int __n, long double __x)

3.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

3.23 std::tr1::__detail Namespace Reference

3.23.1 Detailed Description

Implementation details not part of the namespace std::tr1 interface.

3.24 std::tr2 Namespace Reference

Namespaces

- namespace __detail

Classes

- struct __dynamic_bitset_base
- struct __reflection_typelist< _First, _Rest...>

  Partial specialization.
- struct __reflection_typelist<>

  Specialization for an empty typelist.
- struct bases

  Sequence abstraction metafunctions for manipulating a typelist.
- class bool_set
- struct direct_bases

  Enumerate all the direct base classes of a class. Form of a typelist.
- class dynamic_bitset

  The dynamic_bitset class represents a sequence of bits.

Functions

- bool **certainly** (bool_set __b)
- bool **contains** (bool_set __s, bool_set __t)
- bool **equals** (bool_set __s, bool_set __t)
- bool **is_emptyset** (bool_set __b)
- bool **is_indeterminate** (bool_set __b)
- bool **is_singleton** (bool_set __b)
- bool_set **operator!=** (bool __s, bool_set __t)
- bool_set **operator!=** (bool_set __s, bool __t)
- bool_set **operator!=** (bool_set __s, bool_set __t)
- bool_set **operator&** (bool __s, bool_set __t)
- bool_set **operator&** (bool_set __s, bool __t)
- bool_set **operator==** (bool __s, bool_set __t)
- bool_set **operator==** (bool_set __s, bool __t)
- bool_set **operator^** (bool __s, bool_set __t)
- bool_set **operator^** (bool_set __s, bool __t)
- bool_set **operator|** (bool __s, bool_set __t)
- bool_set **operator|** (bool_set __s, bool __t)
- bool **possibly** (bool_set __b)
- bool_set **set_complement** (bool_set __b)
- bool_set **set_intersection** (bool __s, bool_set __t)
- bool_set **set_intersection** (bool_set __s, bool __t)
- bool_set **set_intersection** (bool_set __s, bool_set __t)
- bool_set **set_union** (bool __s, bool_set __t)
- bool_set **set_union** (bool_set __s, bool __t)
- bool_set **set_union** (bool_set __s, bool_set __t)

- template<typename _WordT , typename _Alloc >
  bool operator== (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT , typename _Alloc >
  bool operator< (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

- template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)
- template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)

3.24.1 Detailed Description

ISO C++ TR2 entities toplevel namespace is std::tr2.

3.24.2 Function Documentation

3.24.2.1 template<typename _WordT , typename _Alloc > bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > & *__lhs*, const dynamic_bitset< _WordT, _Alloc > & *__rhs*)

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1291 of file dynamic_bitset.

3.24.2.2 template<typename _WordT , typename _Alloc > dynamic_bitset<_WordT, _Alloc> std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > & *__x*, const dynamic_bitset< _WordT, _Alloc > & *__y*) [inline]

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1332 of file dynamic_bitset.

3.24.2.3 template<typename _WordT , typename _Alloc > dynamic_bitset<_WordT, _Alloc> std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > & *__x*, const dynamic_bitset< _WordT, _Alloc > & *__y*) `[inline]`

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1362 of file dynamic_bitset.

3.24.2.4 template<typename _WordT , typename _Alloc > bool std::tr2::operator< (const dynamic_bitset< _WordT, _Alloc > & *__lhs*, const dynamic_bitset< _WordT, _Alloc > & *__rhs*)

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1297 of file dynamic_bitset.

3.24.2.5 template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc > std::basic_ostream<_CharT, _Traits>& std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > & *__os*, const dynamic_bitset< _WordT, _Alloc > & *__x*)

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace and only accept '0' and '1' characters. The dynamic_bitset will grow as necessary to hold the string of bits.

Definition at line 1455 of file dynamic_bitset.

References std::__ctype_abstract_base< _CharT >::widen().

3.24.2.6 template<typename _WordT , typename _Alloc > bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > & *__lhs*, const dynamic_bitset< _WordT, _Alloc > & *__rhs*)

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1303 of file dynamic_bitset.

3.24.2.7 template<typename _WordT , typename _Alloc > bool std::tr2::operator== (const dynamic_bitset< _WordT, _Alloc > & *__lhs*, const dynamic_bitset< _WordT, _Alloc > & *__rhs*)

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1285 of file dynamic_bitset.

3.24.2.8 template<typename _WordT , typename _Alloc > bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > & *__lhs*, const dynamic_bitset< _WordT, _Alloc > & *__rhs*)

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1309 of file dynamic_bitset.

3.24.2.9 template<typename _WordT , typename _Alloc > bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > & *__lhs*, const dynamic_bitset< _WordT, _Alloc > & *__rhs*)

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1315 of file dynamic_bitset.

3.24.2.10 template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc > std::basic_istream<_CharT, _Traits>& std::tr2::operator>> (std::basic_istream< _CharT, _Traits > & *__is*, dynamic_bitset< _WordT, _Alloc > & *__x*)

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace and only accept '0' and '1' characters. The dynamic_bitset will grow as necessary to hold the string of bits.

Definition at line 1383 of file dynamic_bitset.

References std::basic_string< _CharT, _Traits, _Alloc >::empty(), std::basic_string< _CharT, _Traits, _Alloc >::push_back(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::tr2::dynamic_bitset< _WordT, _Alloc >::resize(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_string< _CharT, _Traits, _Alloc >::size(), std::tr2::dynamic_bitset< _WordT, _Alloc >::size(), and std::basic_ios< _CharT, _Traits >::widen().

3.24.2.11 template<typename _WordT , typename _Alloc > dynamic_bitset<_WordT, _Alloc> std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > & *__x*, const dynamic_bitset< _WordT, _Alloc > & *__y*) [inline]

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1352 of file dynamic_bitset.

3.24.2.12 template<typename _WordT , typename _Alloc > dynamic_bitset<_WordT, _Alloc> std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > & *__x*, const dynamic_bitset< _WordT, _Alloc > & *__y*) [inline]

Global bitwise operations on bitsets.

Parameters

__x	A bitset.
__y	A bitset of the same size as *__x*.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1342 of file dynamic_bitset.

3.25 std::tr2::__detail Namespace Reference

3.25.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface. Dynamic Bitset.

See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library.

4 Class Documentation

4.1 __cxxabiv1::__forced_unwind Class Reference

4.1.1 Detailed Description

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

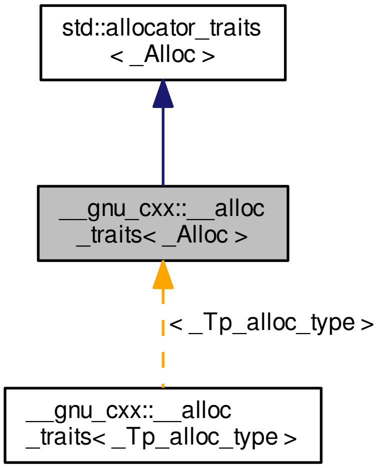
Definition at line 48 of file cxxabi_forced.h.

The documentation for this class was generated from the following file:

- cxxabi_forced.h

4.2 __gnu_cxx::__alloc_traits< _Alloc > Struct Template Reference

Inheritance diagram for __gnu_cxx::__alloc_traits< _Alloc >:



Public Types

- typedef std::allocator_traits
  < _Alloc > **_Base_type**
- typedef _Alloc **allocator_type**
- typedef _Base_type::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef __const_void_pointer const_void_pointer
- typedef _Base_type::difference_type **difference_type**
- typedef _Base_type::pointer **pointer**
- typedef
  __propagate_on_container_copy_assignment propagate_on_container_copy_assignment
- typedef
  __propagate_on_container_move_assignment propagate_on_container_move_assignment
- typedef
  __propagate_on_container_swap propagate_on_container_swap
- template< typename _Tp >
  using **rebind_alloc** = typename __alloctr_rebind< _Alloc, _Tp >::__type
- template< typename _Tp >
  using **rebind_traits** = allocator_traits< rebind_alloc< _Tp > >
- typedef value_type & **reference**
- typedef _Base_type::size_type **size_type**
- typedef _Base_type::value_type **value_type**
- typedef __void_pointer void_pointer

Static Public Member Functions

- static constexpr bool **_S_always_equal** ()
- static constexpr bool **_S_nothrow_move** ()
- static constexpr bool **_S_nothrow_swap** ()
- static void **_S_on_swap** (_Alloc &__a, _Alloc &__b)
- static constexpr bool **_S_propagate_on_copy_assign** ()
- static constexpr bool **_S_propagate_on_move_assign** ()
- static constexpr bool **_S_propagate_on_swap** ()
- static _Alloc **_S_select_on_copy** (const _Alloc &__a)
- static pointer allocate (_Alloc &__a, size_type __n)
- static pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)
- template<typename _Ptr , typename... _Args>
  static std::enable_if
  < __is_custom_pointer< _Ptr >
  ::value >::type **construct** (_Alloc &__a, _Ptr __p, _Args &&...__args)
- template<typename _Tp , typename... _Args>
  static auto construct (_Alloc &__a, _Tp ∗__p, _Args &&...__args) -> decltype(_S_construct(__a, __p, std::forward< _Args >(__args)...))
- static void deallocate (_Alloc &__a, pointer __p, size_type __n)
- template<typename _Ptr >
  static std::enable_if
  < __is_custom_pointer< _Ptr >
  ::value >::type **destroy** (_Alloc &__a, _Ptr __p)
- template<class _Tp >
  static void destroy (_Alloc &__a, _Tp ∗__p)
- static size_type max_size (const _Alloc &__a)
- static _Alloc select_on_container_copy_construction (const _Alloc &__rhs)

4.2.1 Detailed Description

template<typename _Alloc>struct __gnu_cxx::__alloc_traits< _Alloc >

Uniform interface to C++98 and C++0x allocators.

Definition at line 128 of file ext/alloc_traits.h.

4.2.2 Member Typedef Documentation

4.2.2.1 template<typename _Alloc> typedef __const_void_pointer std::allocator_traits< _Alloc >::const_void_pointer `[inherited]`

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 142 of file bits/alloc_traits.h.

4.2.2.2 template<typename _Alloc> typedef __propagate_on_container_copy_assignment std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment `[inherited]`

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 176 of file bits/alloc_traits.h.

4.2.2.3 template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits< _Alloc >::propagate_on_container_move_assignment `[inherited]`

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 188 of file bits/alloc_traits.h.

4.2.2.4 template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits< _Alloc >::propagate_on_container_swap `[inherited]`

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 199 of file bits/alloc_traits.h.

4.2.2.5 template<typename _Alloc> typedef __void_pointer std::allocator_traits< _Alloc >::void_pointer `[inherited]`

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 131 of file bits/alloc_traits.h.

4.2.3 Member Function Documentation

4.2.3.1 template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate (_Alloc & __a, size_type __n) `[inline]`, `[static]`, `[inherited]`

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 352 of file bits/alloc_traits.h.

4.2.3.2 template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate (_Alloc & __a, size_type __n, const_void_pointer __hint) `[inline]`, `[static]`, `[inherited]`

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 367 of file bits/alloc_traits.h.

4.2.3.3 template<typename _Alloc> template<typename _Tp , typename... _Args> static auto **std::allocator_traits**< _Alloc >::construct (_Alloc & *__a*, _Tp * *__p*, _Args &&... *__args*) -> decltype(_S_construct(__a, __p, **std::forward**< _Args>(__args)...)) `[inline]`, `[static]`, `[inherited]`

Construct an object of type *_Tp*.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

Definition at line 393 of file bits/alloc_traits.h.

4.2.3.4 template<typename _Alloc> static void **std::allocator_traits**< _Alloc >::deallocate (_Alloc & *__a*, **pointer** *__p*, **size_type** *__n*) `[inline]`, `[static]`, `[inherited]`

Deallocate memory.

Parameters

__a	An allocator.
__p	Pointer to the memory to deallocate.
__n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 378 of file bits/alloc_traits.h.

4.2.3.5 template<typename _Alloc> template<class _Tp > static void **std::allocator_traits**< _Alloc >::destroy (_Alloc & *__a*, _Tp * *__p*) `[inline]`, `[static]`, `[inherited]`

Destroy an object of type *_Tp*.

Parameters

__a	An allocator.
__p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 406 of file bits/alloc_traits.h.

4.2.3.6 template<typename _Alloc> static size_type std::allocator_traits< _Alloc >::max_size (const _Alloc & *__a*) `[inline]`, `[static]`, `[inherited]`

The maximum supported allocation size.

Parameters

__a	An allocator.

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 417 of file bits/alloc_traits.h.

Referenced by std::forward_list< _Tp, _Alloc >::max_size().

4.2.3.7 template<typename _Alloc> static _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (const _Alloc & *__rhs*) `[inline]`, `[static]`, `[inherited]`

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

Definition at line 429 of file bits/alloc_traits.h.

The documentation for this struct was generated from the following file:

- ext/alloc_traits.h

4.3 __gnu_cxx::__common_pool_policy< _PoolTp, _Thread > Struct Template Reference

Inherits __common_pool_base< _PoolTp, _Thread >.

4.3.1 Detailed Description

template<template< bool > class _PoolTp, bool _Thread>struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >

Policy for shared __pool objects.

Definition at line 460 of file mt_allocator.h.

The documentation for this struct was generated from the following file:

- mt_allocator.h

4.4 __gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference

Public Types

- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef _Tp ∗ **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator __pos) throw ()
- void **insert** (iterator __pos, const_reference __x)
- reference **operator[ ]** (const size_type __pos) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference __x)
- size_type **size** () const throw ()

4.4.1 Detailed Description

template<typename _Tp>class __gnu_cxx::__detail::__mini_vector< _Tp >

__mini_vector<> is a stripped down version of the full-fledged std::vector<>.

It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present. 2. Used ONLY for PODs. 3. No Allocator template argument. Uses operator new() to get memory, and operator delete() to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

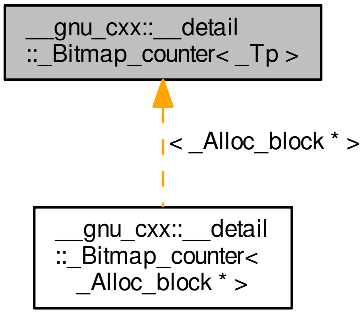
Definition at line 69 of file bitmap_allocator.h.

The documentation for this class was generated from the following file:

- bitmap_allocator.h

4.5 __gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::__detail::_Bitmap_counter< _Tp >:



Public Member Functions

- **_Bitmap_counter** (_BPVector &Rvbp, long __index=-1)
- pointer **_M_base** () const throw ()
- bool **_M_finished** () const throw ()
- size_t ∗ **_M_get** () const throw ()
- _Index_type **_M_offset** () const throw ()
- void **_M_reset** (long __index=-1) throw ()
- void **_M_set_internal_bitmap** (size_t ∗__new_internal_marker) throw ()
- _Index_type **_M_where** () const throw ()
- _Bitmap_counter & **operator++** () throw ()

4.5.1 Detailed Description

template<typename _Tp>class __gnu_cxx::__detail::_Bitmap_counter< _Tp >

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

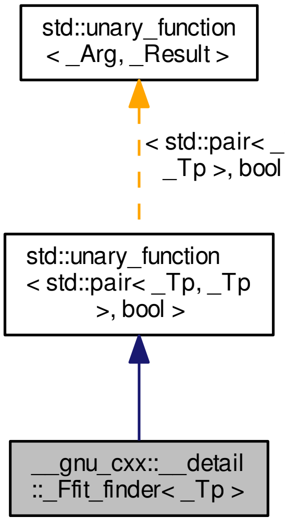
Definition at line 396 of file bitmap_allocator.h.

The documentation for this class was generated from the following file:

- bitmap_allocator.h

4.6 __gnu_cxx::__detail::_Ffit_finder< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::__detail::_Ffit_finder< _Tp >:



Public Types

- typedef std::pair< _Tp, _Tp > argument_type
- typedef bool result_type

Public Member Functions

- size_t ∗ **_M_get** () const throw ()
- _Counter_type **_M_offset** () const throw ()
- bool **operator()** (_Block_pair __bp) throw ()

4.6.1 Detailed Description

template<typename _Tp>class __gnu_cxx::__detail::_Ffit_finder< _Tp >

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 331 of file bitmap_allocator.h.

4.6.2 Member Typedef Documentation

4.6.2.1 typedef **std::pair**< _Tp, _Tp > **std::unary_function**< **std::pair**< _Tp, _Tp > , bool >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.6.2.2 typedef bool **std::unary_function**< **std::pair**< _Tp, _Tp > , bool >**::result_type** `[inherited]`

`result_type` is the return type

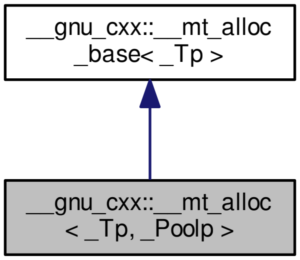
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- bitmap_allocator.h

4.7 __gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference

Inheritance diagram for __gnu_cxx::__mt_alloc< _Tp, _Poolp >:



Public Types

- typedef _Poolp **__policy_type**
- typedef _Poolp::pool_type **__pool_type**
- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__mt_alloc** (const __mt_alloc &) noexcept

- template<typename _Tp1 , typename _Poolp1 >
  __mt_alloc (const __mt_alloc< _Tp1, _Poolp1 > &) noexcept
- const __pool_base::_Tune **_M_get_options** ()
- void **_M_set_options** (__pool_base::_Tune __t)
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void ∗=0)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.7.1 Detailed Description

template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html`.

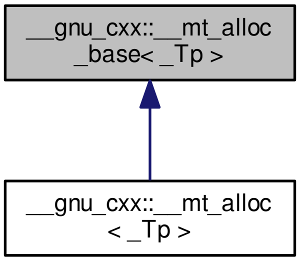
Definition at line 639 of file mt_allocator.h.

The documentation for this class was generated from the following file:

- mt_allocator.h

4.8 __gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::__mt_alloc_base< _Tp >:



Public Types

- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.8.1 Detailed Description

template<typename _Tp>class __gnu_cxx::__mt_alloc_base< _Tp >

Base class for _Tp dependent member functions.

Definition at line 570 of file mt_allocator.h.

The documentation for this class was generated from the following file:

- mt_allocator.h

4.9 __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct Template Reference

Inherits __per_type_pool_base< _Tp, _PoolTp, _Thread >.

4.9.1 Detailed Description

template<typename _Tp, template< bool > class _PoolTp, bool _Thread>struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >

Policy for individual __pool objects.

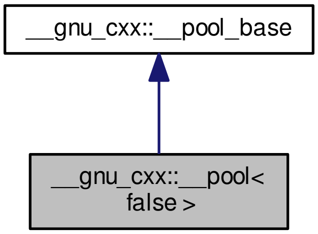
Definition at line 555 of file mt_allocator.h.

The documentation for this struct was generated from the following file:

- mt_allocator.h

4.10 __gnu_cxx::__pool< false > Class Template Reference

Inheritance diagram for __gnu_cxx::__pool< false >:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool** (const __pool_base::_Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &, _Block_record ∗, size_t)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char ∗__p, size_t __bytes) throw ()
- char ∗ **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type ∗ **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

4.10.1 Detailed Description

template<>class __gnu_cxx::__pool< false >

Specialization for single thread.

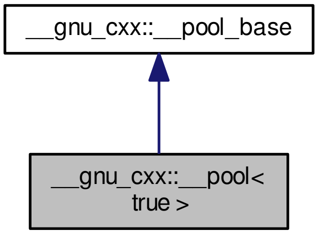
Definition at line 196 of file mt_allocator.h.

The documentation for this class was generated from the following file:

- mt_allocator.h

4.11 __gnu_cxx::__pool< true > Class Template Reference

Inheritance diagram for __gnu_cxx::__pool< true >:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool** (const __pool_base::_Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record ∗__block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- void **_M_destroy_thread_key** (void ∗) throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char ∗__p, size_t __bytes) throw ()
- char ∗ **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type ∗ **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

4.11.1 Detailed Description

template<>class __gnu_cxx::__pool< true >

Specialization for thread enabled, via gthreads.h.

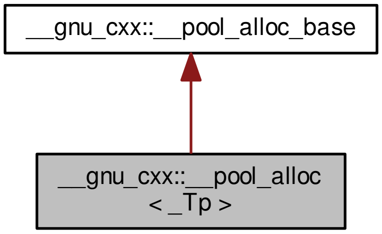
Definition at line 263 of file mt_allocator.h.

The documentation for this class was generated from the following file:

- mt_allocator.h

4.12 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::__pool_alloc< _Tp >:



Public Types

- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__pool_alloc** (const __pool_alloc &) noexcept
- template<typename _Tp1 >
  __pool_alloc (const __pool_alloc< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void ∗=0)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

Private Types

- enum { **_S_align** }
- enum { **_S_max_bytes** }
- enum { **_S_free_list_size** }

Private Member Functions

- char ∗ **_M_allocate_chunk** (size_t __n, int &__nobjs)
- _Obj ∗volatile ∗ **_M_get_free_list** (size_t __bytes) throw ()
- __mutex & **_M_get_mutex** () throw ()
- void ∗ **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Private Attributes

- static char ∗ **_S_end_free**
- static _Obj ∗volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char ∗ **_S_start_free**

4.12.1 Detailed Description

template<typename _Tp>class __gnu_cxx::__pool_alloc< _Tp >

Allocator using a memory pool with a single lock.

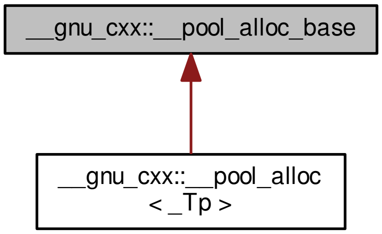
Definition at line 126 of file pool_allocator.h.

The documentation for this class was generated from the following file:

- pool_allocator.h

4.13 __gnu_cxx::__pool_alloc_base Class Reference

Inheritance diagram for __gnu_cxx::__pool_alloc_base:



Protected Types

- enum { **_S_align** }
- enum { **_S_max_bytes** }
- enum { **_S_free_list_size** }

Protected Member Functions

- char ∗ **_M_allocate_chunk** (size_t __n, int &__nobjs)
- _Obj ∗volatile ∗ **_M_get_free_list** (size_t __bytes) throw ()
- __mutex & **_M_get_mutex** () throw ()
- void ∗ **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Protected Attributes

- static char ∗ **_S_end_free**
- static _Obj ∗volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char ∗ **_S_start_free**

4.13.1 Detailed Description

Base class for __pool_alloc.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new 1. If the clients request an object of size $>$ _S_max_bytes, the resulting object will be obtained directly from new 2. In all other cases, we allocate an object of size exactly _S_round_up(requested_size). Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

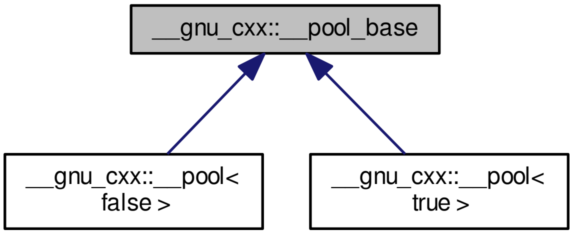
Definition at line 78 of file pool_allocator.h.

The documentation for this class was generated from the following file:

- pool_allocator.h

4.14 __gnu_cxx::__pool_base Struct Reference

Inheritance diagram for __gnu_cxx::__pool_base:



Public Types

- typedef unsigned short int **_Binmap_type**

Public Member Functions

- **__pool_base** (const _Tune &__options)
- bool **_M_check_threshold** (size_t __bytes)
- size_t **_M_get_align** ()
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type ∗ **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

4.14.1 Detailed Description

Base class for pool object.

Definition at line 51 of file mt_allocator.h.

The documentation for this struct was generated from the following file:

- mt_allocator.h

4.15 __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference

Inherits __gnu_cxx::__vstring_utility< _CharT, _Traits, _Alloc >.

Public Types

- typedef
  _Util_Base::_CharT_alloc_type **_CharT_alloc_type**
- typedef __vstring_utility
  < _CharT, _Traits, _Alloc > **_Util_Base**
- typedef _Alloc **allocator_type**
- typedef
  _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **__rc_string_base** (const _Alloc &__a)
- **__rc_string_base** (const __rc_string_base &__rcs)
- **__rc_string_base** (__rc_string_base &&__rcs)
- **__rc_string_base** (size_type __n, _CharT __c, const _Alloc &__a)
- template<typename _InputIterator >
  __rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)
- void **_M_assign** (const __rc_string_base &__rcs)
- size_type **_M_capacity** () const
- void **_M_clear** ()
- bool **_M_compare** (const __rc_string_base &) const
- template<>
  bool **_M_compare** (const __rc_string_base &__rcs) const
- template<>
  bool **_M_compare** (const __rc_string_base &__rcs) const
- _CharT ∗ **_M_data** () const
- void **_M_erase** (size_type __pos, size_type __n)
- allocator_type & **_M_get_allocator** ()
- const allocator_type & **_M_get_allocator** () const
- bool **_M_is_shared** () const
- void **_M_leak** ()
- size_type **_M_length** () const
- size_type **_M_max_size** () const
- void **_M_mutate** (size_type __pos, size_type __len1, const _CharT ∗__s, size_type __len2)
- void **_M_reserve** (size_type __res)
- void **_M_set_leaked** ()
- void **_M_set_length** (size_type __n)
- void **_M_swap** (__rc_string_base &__rcs)
- template<typename _InIterator >
  _CharT ∗ **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward_iterator_tag)

Protected Types

- typedef
  __gnu_cxx::__normal_iterator
  < const_pointer,
  __gnu_cxx::__versa_string
  < _CharT, _Traits, _Alloc,
  __rc_string_base > > **__const_rc_iterator**
- typedef
  __gnu_cxx::__normal_iterator
  < const_pointer,
  __gnu_cxx::__versa_string
  < _CharT, _Traits, _Alloc,
  __sso_string_base > > **__const_sso_iterator**
- typedef
  __gnu_cxx::__normal_iterator
  < pointer,
  __gnu_cxx::__versa_string
  < _CharT, _Traits, _Alloc,
  __rc_string_base > > **__rc_iterator**
- typedef
  __gnu_cxx::__normal_iterator
  < pointer,
  __gnu_cxx::__versa_string
  < _CharT, _Traits, _Alloc,
  __sso_string_base > > **__sso_iterator**
- typedef
  _CharT_alloc_type::const_pointer **const_pointer**
- typedef
  _CharT_alloc_type::difference_type **difference_type**
- typedef _CharT_alloc_type::pointer **pointer**

Static Protected Member Functions

- static void **_S_assign** (_CharT ∗__d, size_type __n, _CharT __c)
- static int **_S_compare** (size_type __n1, size_type __n2)
- static void **_S_copy** (_CharT ∗__d, const _CharT ∗__s, size_type __n)
- template<typename _Iterator >
  static void **_S_copy_chars** (_CharT ∗__p, _Iterator __k1, _Iterator __k2)
- static void **_S_copy_chars** (_CharT ∗__p, __sso_iterator __k1, __sso_iterator __k2)
- static void **_S_copy_chars** (_CharT ∗__p, __const_sso_iterator __k1, __const_sso_iterator __k2)
- static void **_S_copy_chars** (_CharT ∗__p, __rc_iterator __k1, __rc_iterator __k2)
- static void **_S_copy_chars** (_CharT ∗__p, __const_rc_iterator __k1, __const_rc_iterator __k2)
- static void **_S_copy_chars** (_CharT ∗__p, _CharT ∗__k1, _CharT ∗__k2)
- static void **_S_copy_chars** (_CharT ∗__p, const _CharT ∗__k1, const _CharT ∗__k2)
- static void **_S_move** (_CharT ∗__d, const _CharT ∗__s, size_type __n)

4.15.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >

Documentation? What's that? Nathan Myers `ncm@cantrip.org`.

A string looks like this:

```
                                    [_Rep]
                                    _M_length
[__rc_string_base<char_type>]       _M_capacity
_M_dataplus                         _M_refcount
_M_p ---------------->              unnamed array of char_type
```

Where the _M_p points to the first character in the string, and you cast it to a pointer-to-_Rep and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: _Rep::_M_refdata(), and __rc_string_base::_M_rep(); and the allocation function which gets a block of raw bytes and with room enough and constructs a _Rep object at the front.

The reason you want _M_data pointing to the character array and not the _Rep is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the _Rep for the debugger to use, so users can check the actual string length.)

Note that the _Rep object is a POD so that you can have a static *empty string* _Rep object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string _Rep object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 82 of file rc_string_base.h.

The documentation for this class was generated from the following file:

- rc_string_base.h

4.16 __gnu_cxx::__scoped_lock Class Reference

Public Types

- typedef __mutex **__mutex_type**

Public Member Functions

- **__scoped_lock** (__mutex_type &__name)

4.16.1 Detailed Description

Scoped lock idiom.

Definition at line 231 of file concurrence.h.

The documentation for this class was generated from the following file:

- concurrence.h

4.17 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference

Inherits _Base< _CharT, _Traits, _Alloc >.

Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator < const_pointer, __versa_string > **const_iterator**
- typedef _CharT_alloc_type::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _CharT_alloc_type::difference_type **difference_type**
- typedef __gnu_cxx::__normal_iterator < pointer, __versa_string > **iterator**
- typedef _CharT_alloc_type::pointer **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- __versa_string ()
- __versa_string (const _Alloc &__a)
- __versa_string (const __versa_string &__str)
- __versa_string (__versa_string &&__str) noexcept
- __versa_string (std::initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- __versa_string (const __versa_string &__str, size_type __pos, size_type __n=npos)
- __versa_string (const __versa_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- __versa_string (const _CharT ∗__s, size_type __n, const _Alloc &__a=_Alloc())
- __versa_string (const _CharT ∗__s, const _Alloc &__a=_Alloc())
- __versa_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> __versa_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- ∼__versa_string () noexcept
- __versa_string & append (const __versa_string &__str)
- __versa_string & append (const __versa_string &__str, size_type __pos, size_type __n)
- __versa_string & append (const _CharT ∗__s, size_type __n)
- __versa_string & append (const _CharT ∗__s)
- __versa_string & append (size_type __n, _CharT __c)
- __versa_string & append (std::initializer_list< _CharT > __l)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> __versa_string & append (_InputIterator __first, _InputIterator __last)
- __versa_string & assign (const __versa_string &__str)
- __versa_string & assign (__versa_string &&__str)

- __versa_string & assign (const __versa_string &__str, size_type __pos, size_type __n)
- __versa_string & assign (const _CharT ∗__s, size_type __n)
- __versa_string & assign (const _CharT ∗__s)
- __versa_string & assign (size_type __n, _CharT __c)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & assign (_InputIterator __first, _InputIterator __last)
- __versa_string & assign (std::initializer_list< _CharT > __l)
- const_reference at (size_type __n) const
- reference at (size_type __n)
- reference back ()
- const_reference back () const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const _CharT ∗ c_str () const noexcept
- size_type capacity () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- int compare (const __versa_string &__str) const
- int compare (size_type __pos, size_type __n, const __versa_string &__str) const
- int compare (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2) const
- int compare (const _CharT ∗__s) const
- int compare (size_type __pos, size_type __n1, const _CharT ∗__s) const
- int compare (size_type __pos, size_type __n1, const _CharT ∗__s, size_type __n2) const
- size_type copy (_CharT ∗__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- const _CharT ∗ data () const noexcept
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- __versa_string & erase (size_type __pos=0, size_type __n=npos)
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- size_type find (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find (const _CharT ∗__s, size_type __pos=0) const
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_first_not_of (const _CharT ∗__s, size_type __pos=0) const
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_first_of (const _CharT ∗__s, size_type __pos=0) const
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_last_not_of (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_last_not_of (const _CharT ∗__s, size_type __pos=npos) const
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept

- size_type find_last_of (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_last_of (const _CharT ∗__s, size_type __pos=npos) const
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- void insert (iterator __p, size_type __n, _CharT __c)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, std::initializer_list< _CharT > __l)
- __versa_string & insert (size_type __pos1, const __versa_string &__str)
- __versa_string & insert (size_type __pos1, const __versa_string &__str, size_type __pos2, size_type __n)
- __versa_string & insert (size_type __pos, const _CharT ∗__s, size_type __n)
- __versa_string & insert (size_type __pos, const _CharT ∗__s)
- __versa_string & insert (size_type __pos, size_type __n, _CharT __c)
- iterator insert (iterator __p, _CharT __c)
- size_type length () const noexcept
- size_type max_size () const noexcept
- __versa_string & operator+= (const __versa_string &__str)
- __versa_string & operator+= (const _CharT ∗__s)
- __versa_string & operator+= (_CharT __c)
- __versa_string & operator+= (std::initializer_list< _CharT > __l)
- __versa_string & operator= (const __versa_string &__str)
- __versa_string & operator= (__versa_string &&__str)
- __versa_string & operator= (std::initializer_list< _CharT > __l)
- __versa_string & operator= (const _CharT ∗__s)
- __versa_string & operator= (_CharT __c)
- const_reference operator[] (size_type __pos) const
- reference operator[] (size_type __pos)
- void pop_back ()
- void push_back (_CharT __c)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- __versa_string & replace (size_type __pos, size_type __n, const __versa_string &__str)
- __versa_string & replace (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)
- __versa_string & replace (size_type __pos, size_type __n1, const _CharT ∗__s, size_type __n2)
- __versa_string & replace (size_type __pos, size_type __n1, const _CharT ∗__s)
- __versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- __versa_string & replace (iterator __i1, iterator __i2, const __versa_string &__str)
- __versa_string & replace (iterator __i1, iterator __i2, const _CharT ∗__s, size_type __n)
- __versa_string & replace (iterator __i1, iterator __i2, const _CharT ∗__s)
- __versa_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- __versa_string & **replace** (iterator __i1, iterator __i2, _CharT ∗__k1, _CharT ∗__k2)
- __versa_string & **replace** (iterator __i1, iterator __i2, const _CharT ∗__k1, const _CharT ∗__k2)
- __versa_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)

- __versa_string & **replace** (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- __versa_string & replace (iterator __i1, iterator __i2, std::initializer_list< _CharT > __l)
- void reserve (size_type __res_arg=0)
- void resize (size_type __n, _CharT __c)
- void resize (size_type __n)
- size_type rfind (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const
- size_type rfind (const _CharT *__s, size_type __pos=npos) const
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- void shrink_to_fit ()
- size_type size () const noexcept
- __versa_string substr (size_type __pos=0, size_type __n=npos) const
- void swap (__versa_string &__s)

Static Public Attributes

- static const size_type npos

4.17.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

Template class __versa_string.

Data structure managing sequences of characters and character-like objects.

Definition at line 56 of file vstring.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string ( ) [inline]

Default constructor creates an empty string.

Definition at line 134 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr().

4.17.2.2 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (const _Alloc & *__a*) [inline], [explicit]

Construct an empty string using allocator *a*.

Definition at line 141 of file vstring.h.

4.17.2.3 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) [inline]

Construct string with copy of value of *__str*.

Parameters

__str	Source string.

Definition at line 149 of file vstring.h.

4.17.2.4 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (__versa_string< _CharT, _Traits, _Alloc, _Base > && *__str*) [inline], [noexcept]

String move constructor.

Parameters

__str	Source string.

The newly-constructed string contains the exact contents of *__str*. The contents of *__str* are a valid, but unspecified string.

Definition at line 161 of file vstring.h.

4.17.2.5 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (std::initializer_list< _CharT > *__l*, const _Alloc & *__a* = _Alloc()) [inline]

Construct string from an initializer list.

Parameters

__l	std::initializer_list of characters.
__a	Allocator to use (default is default allocator).

Definition at line 169 of file vstring.h.

4.17.2.6 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos*, size_type *__n* = npos) [inline]

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy (default remainder).

Definition at line 180 of file vstring.h.

4.17.2.7 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos*, size_type *__n*, const _Alloc & *__a*) [inline]

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

Definition at line 195 of file vstring.h.

4.17.2.8 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (const _CharT * *__s*, size_type *__n*, const _Alloc & *__a* = `_Alloc()`) `[inline]`

Construct string initialized by a character array.

Parameters

__s	Source character array.
__n	Number of characters to copy.
__a	Allocator to use (default is default allocator).

NB: *__s* must have at least *__n* characters, '\0' has no special meaning.

Definition at line 212 of file vstring.h.

4.17.2.9 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (const _CharT * *__s*, const _Alloc & *__a* = `_Alloc()`) `[inline]`

Construct string as copy of a C string.

Parameters

__s	Source C string.
__a	Allocator to use (default is default allocator).

Definition at line 221 of file vstring.h.

4.17.2.10 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (size_type *__n*, _CharT *__c*, const _Alloc & *__a* = `_Alloc()`) `[inline]`

Construct string as multiple characters.

Parameters

__n	Number of characters.
__c	Character to use.
__a	Allocator to use (default is default allocator).

Definition at line 231 of file vstring.h.

4.17.2.11 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (_InputIterator *__beg*, _InputIterator *__end*, const _Alloc & *__a* = `_Alloc()`) `[inline]`

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

Definition at line 246 of file vstring.h.

4.17.2.12 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::∼__versa_string ( ) `[inline]`, `[noexcept]`

Destroy the string instance.

Definition at line 253 of file vstring.h.

4.17.3 Member Function Documentation

4.17.3.1 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) `[inline]`

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Definition at line 688 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by std::getline(), __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=().

4.17.3.2 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str,* size_type *__pos,* size_type *__n*) `[inline]`

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

Definition at line 705 of file vstring.h.

4.17.3.3 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const _CharT * *__s,* size_type *__n*) [inline]

Append a C substring.

Parameters

__s	The C string to append.
__n	The number of characters to append.

Returns

Reference to this string.

Definition at line 717 of file vstring.h.

4.17.3.4 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (const _CharT * *__s*) [inline]

Append a C string.

Parameters

__s	The C string to append.

Returns

Reference to this string.

Definition at line 730 of file vstring.h.

4.17.3.5 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (size_type *__n,* _CharT *__c*) [inline]

Append multiple characters.

Parameters

__n	The number of characters to append.
__c	The character to use.

Returns

Reference to this string.

Appends n copies of c to this string.

Definition at line 747 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.6 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (std::initializer_list< _CharT > *__l*) [inline]

Append an initializer_list of characters.

Parameters

__l	The initializer_list of characters to append.

Returns

Reference to this string.

Definition at line 757 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append().

4.17.3.7 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (_InputIterator *__first,* _InputIterator *__last*) [inline]

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 776 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.8 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) [inline]

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Definition at line 799 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator=().

4.17.3.9 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (__versa_string< _CharT, _Traits, _Alloc, _Base > && *__str*) [inline]

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

Definition at line 815 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap().

4.17.3.10 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos*, size_type *__n*) [inline]

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

Definition at line 836 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.11 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> **__versa_string& __gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::assign (const _CharT * *__s*, size_type *__n*) `[inline]`

Set value to a C substring.

Parameters

__s	The C string to use.
__n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *__n* characters of *__s*. If *__n* is is larger than the number of available characters in *__s*, the remainder of *__s* is used.

Definition at line 853 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.12 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> **__versa_string& __gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::assign (const _CharT * *__s*) `[inline]`

Set value to contents of a C string.

Parameters

__s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *__s*. The data is copied, so there is no dependence on *__s* once the function returns.

Definition at line 869 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.13 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> **__versa_string& __gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::assign (size_type *__n*, _CharT *__c*) `[inline]`

Set value to multiple characters.

Parameters

__n	Length of the resulting string.
__c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to *__n* copies of character *__c*.

Definition at line 886 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.14 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (_InputIterator *__first*, _InputIterator *__last*) [inline]

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 905 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.15 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (std::initializer_list< _CharT > *__l*) [inline]

Set value to an initializer_list of characters.

Parameters

__l	The initializer_list of characters to assign.

Returns

Reference to this string.

Definition at line 915 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign().

4.17.3.16 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (size_type *__n*) const [inline]

Provides access to the data contained in the string.

Parameters

__n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

Definition at line 579 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.17 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (size_type __n) [inline]

Provides access to the data contained in the string.

Parameters

__n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

Definition at line 598 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.18 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ( ) [inline]

Returns a read/write reference to the data at the last element of the string.

Definition at line 628 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[](), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.19 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ( ) const [inline]

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 636 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[](), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.20 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 319 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend().

4.17.3.21 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 330 of file vstring.h.

4.17.3.22 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT∗ __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str ( ) const [inline], [noexcept]

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1538 of file vstring.h.

4.17.3.23 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity ( ) const [inline], [noexcept]

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 490 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit().

4.17.3.24 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 394 of file vstring.h.

4.17.3.25 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 402 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.26 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear ( ) [inline], [noexcept]

Erases the string, making it empty.

Definition at line 518 of file vstring.h.

4.17.3.27 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) const [inline]

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1964 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), std::min(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::operator<(), __gnu_cxx::operator<=(), __gnu_cxx::operator==(), __gnu_cxx::operator>(), and __gnu_cxx::operator>=().

4.17.3.28 template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (size_type *__pos,* size_type *__n,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) const

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 459 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), std::min(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.29 template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (size_type *__pos1*, size_type *__n1*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos2*, size_type *__n2*) const

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 476 of file vstring.tcc.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), and std::min().

4.17.3.30 template< typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (const _CharT * *__s*) const

Compare to a C string.

Parameters

__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__s*, 0 if their values are equivalent, or > 0 if this string is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from *__s*. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 495 of file vstring.tcc.

References std::min(), and std::size().

4.17.3.31 template< typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (size_type *__pos,* size_type *__n1,* const _CharT * *__s*) const

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Returns an integer $<$ 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or $>$ 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 511 of file vstring.tcc.

References std::min().

4.17.3.32 template< typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (size_type *__pos,* size_type *__n1,* const _CharT * *__s,* size_type *__n2*) const

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer $<$ 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or $>$ 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: __s must have at least n2 characters, *\0* has no special meaning.

Definition at line 528 of file vstring.tcc.

References std::min().

4.17.3.33 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::copy (_CharT * __s, size_type __n, size_type __pos = 0) const

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *s*. If *__pos* is greater than size(), out_of_range is thrown.

Definition at line 254 of file vstring.tcc.

4.17.3.34 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 411 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end().

4.17.3.35 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 420 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin().

4.17.3.36 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data ( ) const [inline], [noexcept]

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1548 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of().

4.17.3.37 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty ( ) const [inline], [noexcept]

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 526 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.38 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 338 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin().

4.17.3.39 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 349 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.40 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (size_type *__pos* = 0, size_type *__n* = npos) [inline]

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1115 of file vstring.h.

Referenced by std::getline().

4.17.3.41 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (iterator *__position*) [inline]

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1131 of file vstring.h.

4.17.3.42 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (iterator *__first,* iterator *__last*) [inline]

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1152 of file vstring.h.

4.17.3.43 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (const _CharT * *__s,* size_type *__pos,* size_type *__n*) const

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *__s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 269 of file vstring.tcc.

References std::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of().

4.17.3.44 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos* = 0) const `[inline]`, `[noexcept]`

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1584 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find().

4.17.3.45 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (const _CharT * *__s*, size_type *__pos* = 0) const `[inline]`

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1599 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find().

4.17.3.46 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (_CharT *__c*, size_type *__pos* = 0) const `[noexcept]`

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 293 of file vstring.tcc.

References std::size().

4.17.3.47 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos* = 0) const [inline], [noexcept]

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1816 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of().

4.17.3.48 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (const _CharT * *__s*, size_type *__pos*, size_type *__n*) const

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 391 of file vstring.tcc.

References std::size().

4.17.3.49 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (const _CharT * *__s*, size_type *__pos* = 0) const [inline]

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1847 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of().

4.17.3.50 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (_CharT *__c*, size_type *__pos* = 0) const [noexcept]

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 404 of file vstring.tcc.

References std::size().

4.17.3.51 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos* = 0) const [inline], [noexcept]

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1689 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of().

4.17.3.52 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (const _CharT ∗ *__s*, size_type *__pos*, size_type *__n*) const

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 352 of file vstring.tcc.

References std::size().

4.17.3.53 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (const _CharT * __s, size_type __pos = 0) const [inline]

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1719 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of().

4.17.3.54 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (_CharT __c, size_type __pos = 0) const [inline], [noexcept]

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Note: equivalent to find(c, pos).

Definition at line 1738 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find().

4.17.3.55 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str, size_type __pos = npos) const [inline], [noexcept]

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1879 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of().

4.17.3.56 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 416 of file vstring.tcc.

References std::size().

4.17.3.57 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (const _CharT * __s, size_type __pos = npos) const [inline]

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1910 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of().

4.17.3.58 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (_CharT *__c*, size_type *__pos* = npos) const [noexcept]

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 438 of file vstring.tcc.

References std::size().

4.17.3.59 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos* = npos) const [inline], [noexcept]

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1753 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of().

4.17.3.60 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (const _CharT * *__s*, size_type *__pos*, size_type *__n*) const

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 369 of file vstring.tcc.

References std::size().

4.17.3.61 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (const _CharT * *__s*, size_type *__pos* = npos) const [inline]

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1783 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of().

4.17.3.62 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (_CharT *__c*, size_type *__pos* = npos) const [inline], [noexcept]

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Note: equivalent to rfind(c, pos).

Definition at line 1802 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind().

4.17.3.63 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front ( ) `[inline]`

Returns a read/write reference to the data at the first element of the string.

Definition at line 612 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]().

4.17.3.64 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front ( ) const `[inline]`

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 620 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[]().

4.17.3.65 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ( ) const `[inline]`, `[noexcept]`

Return copy of allocator used to construct this string.

Definition at line 1555 of file vstring.h.

4.17.3.66 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (iterator *__p*, size_type *__n*, _CharT *__c*) `[inline]`

Insert multiple characters.

Parameters

__p	Iterator referencing location in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 933 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.67 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (iterator *__p*, _InputIterator *__beg*, _InputIterator *__end*) `[inline]`

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 955 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.68 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void **__gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::insert (iterator *__p*, **std::initializer_list**< _CharT > *__l*) `[inline]`

Insert an initializer_list of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Definition at line 966 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert().

4.17.3.69 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> **__versa_string**& **__gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::insert (size_type *__pos1*, const **__versa_string**< _CharT, _Traits, _Alloc, _Base > & *__str*) `[inline]`

Insert value of a string.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 983 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.70 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (size_type *__pos1*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos2*, size_type *__n*) [inline]

Insert a substring.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos1* > size() or *__pos2* > *__str.size()*.

Starting at *__pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1006 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.71 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (size_type *__pos*, const _CharT * *__s*, size_type *__n*) [inline]

Insert a C substring.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1029 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.72 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> **__versa_string& __gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::insert (size_type *__pos*, const _CharT * *__s*) `[inline]`

Insert a C string.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1048 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.73 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> **__versa_string& __gnu_cxx::__versa_string**< _CharT, _Traits, _Alloc, _Base >::insert (size_type *__pos*, size_type *__n*, _CharT *__c*) `[inline]`

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1072 of file vstring.h.

4.17.3.74 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (iterator *__p*, _CharT *__c*) `[inline]`

Insert one character.

Parameters

__p	Iterator referencing position in string to insert at.
__c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1090 of file vstring.h.

4.17.3.75 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length ( ) const `[inline]`, `[noexcept]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 435 of file vstring.h.

4.17.3.76 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size ( ) const `[inline]`, `[noexcept]`

Returns the size() of the largest possible string.

Definition at line 440 of file vstring.h.

Referenced by std::getline().

4.17.3.77 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) [inline]

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Definition at line 647 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append().

4.17.3.78 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (const _CharT ∗ *__s*) [inline]

Append a C string.

Parameters

__s	The C string to append.

Returns

Reference to this string.

Definition at line 656 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append().

4.17.3.79 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (_CharT *__c*) [inline]

Append a character.

Parameters

__c	The character to append.

Returns

Reference to this string.

Definition at line 665 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back().

4.17.3.80 template< typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (std::initializer_list< _CharT > __l) [inline]

Append an initializer_list of characters.

Parameters

__l	The initializer_list of characters to be appended.

Returns

Reference to this string.

Definition at line 678 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append().

4.17.3.81 template< typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]

Assign the value of *str* to this string.

Parameters

__str	Source string.

Definition at line 260 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign().

4.17.3.82 template< typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (__versa_string< _CharT, _Traits, _Alloc, _Base > && __str) [inline]

String move assignment operator.

Parameters

__str	Source string.

The contents of *__str* are moved into this string (without copying). *__str* is a valid, but unspecified string.

Definition at line 272 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap().

4.17.3.83 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (std::initializer_list< _CharT > __l) [inline]

Set value to string constructed from initializer list.

Parameters

__l	std::initializer_list.

Definition at line 284 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign().

4.17.3.84 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (const _CharT ∗ __s) [inline]

Copy contents of *__s* into this string.

Parameters

__s	Source null-terminated string.

Definition at line 296 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign().

4.17.3.85 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (_CharT __c) [inline]

Set value to string of length 1.

Parameters

__c	Source character.

Assigning to a character makes this string length 1 and (∗this)[0] == *__c*.

Definition at line 307 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign().

4.17.3.86 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (size_type __pos) const [inline]

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 541 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front().

4.17.3.87 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (size_type __pos) [inline]

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().) Unshares the string.

Definition at line 558 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.88 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back ( ) [inline]

Remove the last character.

The string must be non-empty.

Definition at line 1169 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.89 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back (_CharT __c) [inline]

Append a single character.

Parameters

__c	Character to append.

Definition at line 784 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=().

4.17.3.90 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 358 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end().

4.17.3.91 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 367 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end().

4.17.3.92 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 376 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin().

4.17.3.93 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 385 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin().

4.17.3.94 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type *__pos*, size_type *__n*, const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) [inline]

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos+n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1191 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.95 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type *__pos1,* size_type *__n1,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str,* size_type *__pos2,* size_type *__n2*) `[inline]`

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos1,pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1214 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.96 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type *__pos,* size_type *__n1,* const _CharT ∗ *__s,* size_type *__n2*) [inline]

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *__s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1242 of file vstring.h.

4.17.3.97 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type *__pos,* size_type *__n1,* const _CharT ∗ *__s*) [inline]

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, the characters of *__s* are inserted. If *pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1266 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.98 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (size_type *__pos,* size_type *__n1,* size_type *__n2,* _CharT *__c*) `[inline]`

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1290 of file vstring.h.

4.17.3.99 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator *__i1,* iterator *__i2,* const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*) `[inline]`

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1308 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.100 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator *__i1*, iterator *__i2*, const _CharT * *__s*, size_type *__n*) `[inline]`

Replace range of characters with C substring.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.
__n	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the first *n* characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1326 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.101 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator *__i1*, iterator *__i2*, const _CharT * *__s*) `[inline]`

Replace range of characters with C string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1347 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.102 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator *__i1*, iterator *__i2*, size_type *__n*, _CharT *__c*) `[inline]`

Replace range of characters with multiple characters.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__n	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, *__n* copies of *__c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1368 of file vstring.h.

4.17.3.103 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator *__i1*, iterator *__i2*, _InputIterator *__k1*, _InputIterator *__k2*) `[inline]`

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1393 of file vstring.h.

4.17.3.104 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (iterator *__i1*, iterator *__i2*, std::initializer_list< _CharT > *__l*) `[inline]`

Replace range of characters with initializer_list.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1474 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace().

4.17.3.105 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve (size_type *__res_arg* = `0`) `[inline]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 511 of file vstring.h.

Referenced by __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit().

4.17.3.106 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (size_type *__n*, _CharT *__c*)

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.
__c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *__c*.

Definition at line 50 of file vstring.tcc.

References std::size().

4.17.3.107 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (size_type *__n*) [inline]

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 467 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize().

4.17.3.108 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (const __versa_string< _CharT, _Traits, _Alloc, _Base > & *__str*, size_type *__pos* = npos) const [inline], [noexcept]

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1629 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind().

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind().

4.17.3.109 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (const _CharT * __s, size_type __pos, size_type __n) const

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 312 of file vstring.tcc.

References std::min(), and std::size().

4.17.3.110 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (const _CharT * __s, size_type __pos = npos) const [inline]

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1659 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind().

4.17.3.111 template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (_CharT __c, size_type __pos = npos) const [noexcept]

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 334 of file vstring.tcc.

References std::size().

4.17.3.112 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit ( ) [inline]

A non-binding request to reduce capacity() to size().

Definition at line 473 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

4.17.3.113 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size ( ) const [inline], [noexcept]

Returns the number of characters in the string, not including any null-termination.

Definition at line 429 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert(), __gnu_cxx::operator+(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[](), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit().

4.17.3.114 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr (size_type *__pos* = 0, size_type *__n* = npos) const [inline]

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

Definition at line 1943 of file vstring.h.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string().

4.17.3.115 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > & __s) [inline]

Swap contents with another string.

Parameters

__s	String to swap with.

Exchanges the contents of this string with that of *__s* in constant time.

Definition at line 1527 of file vstring.h.

Referenced by __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator=(), and __gnu_cxx::swap().

4.17.4 Member Data Documentation

4.17.4.1 template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos [static]

Value returned by various member functions when they fail.

Definition at line 81 of file vstring.h.

The documentation for this class was generated from the following files:

- vstring.h
- vstring.tcc

4.18 __gnu_cxx::_Caster< _ToType > Struct Template Reference

Public Types

- typedef _ToType::element_type * **type**

4.18.1 Detailed Description

template<typename _ToType>struct __gnu_cxx::_Caster< _ToType >

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the apprpriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef element_type which names tehe type it points to. 2) has a get() const method which returns element_type∗. 3) has a constructor which can take one element_type∗ argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file cast.h.

The documentation for this struct was generated from the following file:

- cast.h

4.19 __gnu_cxx::_Char_types< _CharT > Struct Template Reference

Public Types

- typedef unsigned long **int_type**
- typedef std::streamoff **off_type**
- typedef std::streampos **pos_type**
- typedef std::mbstate_t **state_type**

4.19.1 Detailed Description

template<typename _CharT>struct __gnu_cxx::_Char_types< _CharT >

Mapping from character type to associated types.

Note

> This is an implementation class for the generic version of char_traits. It defines int_type, off_type, pos_type, and state_type. By default these are unsigned long, streamoff, streampos, and mbstate_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in char_traits, can specialize __gnu_cxx::_Char_types while leaving __gnu_cxx::char_traits alone.

Definition at line 58 of file char_traits.h.

The documentation for this struct was generated from the following file:

- char_traits.h

4.20 __gnu_cxx::_ExtPtr_allocator< _Tp > Class Template Reference

Public Types

- typedef _Pointer_adapter
  < _Relative_pointer_impl
  < const _Tp > > **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Pointer_adapter
  < _Relative_pointer_impl< _Tp > > **pointer**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_ExtPtr_allocator** (const _ExtPtr_allocator &__rarg) noexcept
- template<typename _Up >
  _ExtPtr_allocator (const _ExtPtr_allocator< _Up > &__rarg) noexcept
- const std::allocator< _Tp > & **_M_getUnderlyingImp** () const
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, void ∗__hint=0)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- template<typename... _Args>
  void **construct** (pointer __p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- void **destroy** (pointer __p)
- size_type **max_size** () const noexcept
- template<typename _Up >
  bool **operator!=** (const _ExtPtr_allocator< _Up > &__rarg)
- bool **operator!=** (const _ExtPtr_allocator &__rarg)
- template<typename _Up >
  bool **operator==** (const _ExtPtr_allocator< _Up > &__rarg)
- bool **operator==** (const _ExtPtr_allocator &__rarg)

Friends

- template<typename _Up >
  void **swap** (_ExtPtr_allocator< _Up > &, _ExtPtr_allocator< _Up > &)

4.20.1 Detailed Description

template<typename _Tp>class __gnu_cxx::_ExtPtr_allocator< _Tp >

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.

Definition at line 56 of file extptr_allocator.h.

The documentation for this class was generated from the following file:

- extptr_allocator.h

4.21 __gnu_cxx::_Invalid_type Struct Reference

4.21.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize _Pointer_adapter for cases of void∗, const void∗, and so on.

Definition at line 213 of file pointer.h.

The documentation for this struct was generated from the following file:

- pointer.h

4.22 __gnu_cxx::_Pointer_adapter< _Storage_policy > Class Template Reference

Inherits _Storage_policy.

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef
  _Storage_policy::element_type **element_type**
- typedef
  std::random_access_iterator_tag **iterator_category**
- typedef _Pointer_adapter **pointer**
- typedef _Reference_type
  < element_type >::reference **reference**
- typedef _Unqualified_type
  < element_type >::type **value_type**

Public Member Functions

- **_Pointer_adapter** (element_type ∗__arg=0)
- **_Pointer_adapter** (const _Pointer_adapter &__arg)
- template<typename _Up >
  _Pointer_adapter (_Up ∗__arg)
- template<typename _Up >
  _Pointer_adapter (const _Pointer_adapter< _Up > &__arg)
- **operator __unspecified_bool_type** () const
- bool **operator!** () const
- reference **operator∗** () const
- _Pointer_adapter & **operator++** ()
- _Pointer_adapter **operator++** (int)
- _Pointer_adapter & **operator+=** (short __offset)
- _Pointer_adapter & **operator+=** (unsigned short __offset)
- _Pointer_adapter & **operator+=** (int __offset)
- _Pointer_adapter & **operator+=** (unsigned int __offset)
- _Pointer_adapter & **operator+=** (long __offset)
- _Pointer_adapter & **operator+=** (unsigned long __offset)
- template<typename _Up >
  std::ptrdiff_t **operator-** (const _Pointer_adapter< _Up > &__rhs) const
- _Pointer_adapter & **operator--** ()
- _Pointer_adapter **operator--** (int)
- _Pointer_adapter & **operator-=** (short __offset)
- _Pointer_adapter & **operator-=** (unsigned short __offset)
- _Pointer_adapter & **operator-=** (int __offset)
- _Pointer_adapter & **operator-=** (unsigned int __offset)
- _Pointer_adapter & **operator-=** (long __offset)
- _Pointer_adapter & **operator-=** (unsigned long __offset)

- element_type ∗ **operator->** () const
- _Pointer_adapter & **operator=** (const _Pointer_adapter &__arg)
- template<typename _Up >
  _Pointer_adapter & **operator=** (const _Pointer_adapter< _Up > &__arg)
- template<typename _Up >
  _Pointer_adapter & **operator=** (_Up ∗__arg)
- reference **operator[ ]** (std::ptrdiff_t __index) const

Friends

- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, short __offset)
- _Pointer_adapter **operator+** (short __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned short __offset)
- _Pointer_adapter **operator+** (unsigned short __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, int __offset)
- _Pointer_adapter **operator+** (int __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned int __offset)
- _Pointer_adapter **operator+** (unsigned int __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, long __offset)
- _Pointer_adapter **operator+** (long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned long __offset)
- std::ptrdiff_t **operator-** (const _Pointer_adapter &__lhs, element_type ∗__rhs)
- std::ptrdiff_t **operator-** (element_type ∗__lhs, const _Pointer_adapter &__rhs)
- template<typename _Up >
  std::ptrdiff_t **operator-** (const _Pointer_adapter &__lhs, _Up ∗__rhs)
- template<typename _Up >
  std::ptrdiff_t **operator-** (_Up ∗__lhs, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, short __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned short __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, int __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned int __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned long __offset)

4.22.1 Detailed Description

template<typename _Storage_policy>class __gnu_cxx::_Pointer_adapter< _Storage_policy >

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' Allocator::pointer is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the Allocator::pointer typedef appropriately for pointer types. 2) if you need pointer casting, use the __pointer_cast<> functions from ext/cast.h. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

_Tp∗ == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp∗ == _Pointer_adapter<_Std_pointer_impl<const _Tp> >; _Tp∗ const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp∗ const == const _Pointer_-adapter<_Std_pointer_impl<const _Tp> >;

Definition at line 281 of file pointer.h.

The documentation for this class was generated from the following file:

- pointer.h

4.23 __gnu_cxx::_Relative_pointer_impl< _Tp > Class Template Reference

Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp ∗ **get** () const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const
- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- void **set** (_Tp ∗__arg)

4.23.1 Detailed Description

template<typename _Tp>class __gnu_cxx::_Relative_pointer_impl< _Tp >

A storage policy for use with _Pointer_adapter<> which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 109 of file pointer.h.

The documentation for this class was generated from the following file:

- pointer.h

4.24 __gnu_cxx::_Relative_pointer_impl< const _Tp > Class Template Reference

Public Types

- typedef const _Tp **element_type**

Public Member Functions

- const _Tp ∗ **get** () const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const

- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- void **set** (const _Tp ∗__arg)

4.24.1 Detailed Description

template<typename _Tp>class __gnu_cxx::_Relative_pointer_impl< const _Tp >

Relative_pointer_impl needs a specialization for const T because of the casting done during pointer arithmetic.

Definition at line 161 of file pointer.h.

The documentation for this class was generated from the following file:

- pointer.h

4.25 __gnu_cxx::_Std_pointer_impl< _Tp > Class Template Reference

Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp ∗ **get** () const
- bool **operator<** (const _Std_pointer_impl &__rarg) const
- bool **operator==** (const _Std_pointer_impl &__rarg) const
- void **set** (element_type ∗__arg)

4.25.1 Detailed Description

template<typename _Tp>class __gnu_cxx::_Std_pointer_impl< _Tp >

A storage policy for use with _Pointer_adapter<> which yields a standard pointer.

A _Storage_policy is required to provide 4 things: 1) A get() API for returning the stored pointer value. 2) An set() API for storing a pointer value. 3) An element_type typedef to define the type this points to. 4) An operator<() to support pointer comparison. 5) An operator==() to support pointer comparison.

Definition at line 66 of file pointer.h.

The documentation for this class was generated from the following file:

- pointer.h

4.26 __gnu_cxx::_Unqualified_type< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

4.26.1 Detailed Description

template<typename _Tp>struct __gnu_cxx::_Unqualified_type< _Tp >

This structure accommodates the way in which std::iterator_traits<> is normally specialized for const T∗, so that value_type is still T.

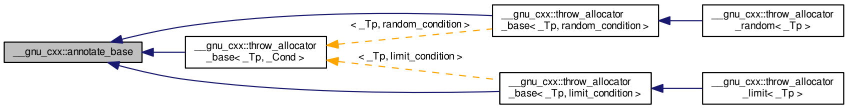
Definition at line 241 of file pointer.h.

The documentation for this struct was generated from the following file:

- pointer.h

4.27 __gnu_cxx::annotate_base Struct Reference

Inheritance diagram for __gnu_cxx::annotate_base:



Public Member Functions

- void **check_allocated** (void ∗p, size_t size)
- void **check_allocated** (size_t label)
- void **erase** (void ∗p, size_t size)
- void **insert** (void ∗p, size_t size)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- std::ostream & **operator**<< (std::ostream &, const annotate_base &)

4.27.1 Detailed Description

Base class for checking address and label information about allocations. Create a std::map between the allocated address (void∗) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

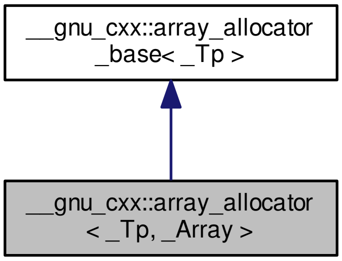
Definition at line 88 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.28 __gnu_cxx::array_allocator< _Tp, _Array > Class Template Reference

Inheritance diagram for __gnu_cxx::array_allocator< _Tp, _Array >:



Public Types

- typedef _Array **array_type**
- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **array_allocator** (array_type ∗__array=0) noexcept
- **array_allocator** (const array_allocator &__o) noexcept
- template< typename _Tp1 , typename _Array1 >
  array_allocator (const array_allocator< _Tp1, _Array1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void ∗=0)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer, size_type)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.28.1 Detailed Description

template<typename _Tp, typename _Array = std::tr1::array<_Tp, 1>>class __gnu_cxx::array_allocator< _Tp, _Array >

An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.

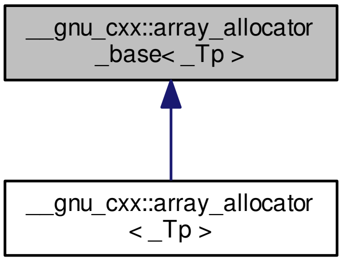
Definition at line 106 of file array_allocator.h.

The documentation for this class was generated from the following file:

- array_allocator.h

4.29 __gnu_cxx::array_allocator_base< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::array_allocator_base< _Tp >:



Public Types

- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer, size_type)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.29.1 Detailed Description

template<typename _Tp>class __gnu_cxx::array_allocator_base< _Tp >

Base class.

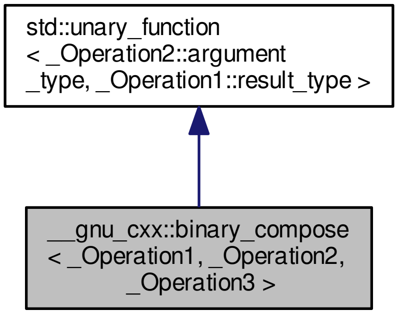
Definition at line 50 of file array_allocator.h.

The documentation for this class was generated from the following file:

- array_allocator.h

4.30 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference

Inheritance diagram for __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

Public Member Functions

- **binary_compose** (const _Operation1 &__x, const _Operation2 &__y, const _Operation3 &__z)
- _Operation1::result_type **operator()** (const typename _Operation2::argument_type &__x) const

Protected Attributes

- _Operation1 **_M_fn1**
- _Operation2 **_M_fn2**
- _Operation3 **_M_fn3**

4.30.1 Detailed Description

template<class _Operation1, class _Operation2, class _Operation3>class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >

An SGI extension .

Definition at line 150 of file ext/functional.

4.30.2 Member Typedef Documentation

4.30.2.1 template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.30.2.2 template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type [inherited]

`result_type` is the return type

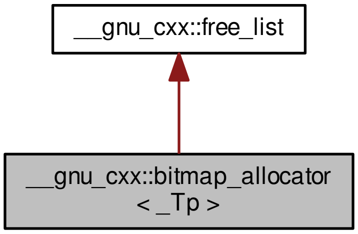
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- ext/functional

4.31 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::bitmap_allocator< _Tp >:



Public Types

- typedef free_list::__mutex_type **__mutex_type**
- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**

- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **bitmap_allocator** (const bitmap_allocator &) noexcept
- template<typename _Tp1 >
  bitmap_allocator (const bitmap_allocator< _Tp1 > &) noexcept
- pointer _M_allocate_single_object () throw (std::bad_alloc)
- void _M_deallocate_single_object (pointer __p) throw ()
- pointer **address** (reference __r) const noexcept
- const_pointer **address** (const_reference __r) const noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, typename bitmap_allocator< void >::const_pointer)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n) throw ()
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

Private Types

- typedef vector_type::iterator **iterator**
- typedef
  __detail::__mini_vector
  < value_type > **vector_type**

Private Member Functions

- void _M_clear ()
- size_t ∗ _M_get (size_t __sz) throw (std::bad_alloc)
- void _M_insert (size_t ∗__addr) throw ()

4.31.1 Detailed Description

template<typename _Tp>class __gnu_cxx::bitmap_allocator< _Tp >

Bitmap Allocator, primary template.

Definition at line 687 of file bitmap_allocator.h.

4.31.2 Member Function Documentation

4.31.2.1 template<typename _Tp > pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object ( ) throw (std::bad_alloc) [inline]

Allocates memory for a single object of size sizeof(_Tp).

Exceptions

std::bad_alloc.	If memory can not be allocated.

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 827 of file bitmap_allocator.h.

References __gnu_cxx::__detail::__bit_allocate(), __gnu_cxx::__detail::__num_bitmaps(), and __gnu_cxx::_Bit_scan_forward().

4.31.2.2 template<typename _Tp > void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (pointer *__p*) throw () [inline]

Deallocates memory that belongs to a single object of size sizeof(_Tp).

Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.

Definition at line 917 of file bitmap_allocator.h.

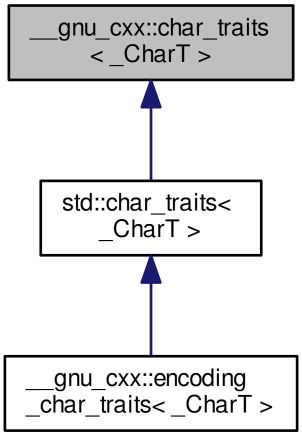
References __gnu_cxx::__detail::__bit_free(), __gnu_cxx::__detail::__num_bitmaps(), std::__rotate(), and __gnu_cxx::free_list::_M_insert().

The documentation for this class was generated from the following file:

- bitmap_allocator.h

4.32 __gnu_cxx::char_traits< _CharT > Struct Template Reference

Inheritance diagram for __gnu_cxx::char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- typedef _Char_types< _CharT > ::int_type **int_type**
- typedef _Char_types< _CharT > ::off_type **off_type**
- typedef _Char_types< _CharT > ::pos_type **pos_type**
- typedef _Char_types< _CharT > ::state_type **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type ∗ **assign** (char_type ∗__s, std::size_t __n, char_type __a)
- static int **compare** (const char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static char_type ∗ **copy** (char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type ∗ **find** (const char_type ∗__s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type ∗__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type ∗ **move** (char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)

- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

4.32.1 Detailed Description

template<typename _CharT>struct __gnu_cxx::char_traits< _CharT >

Base class used to implement std::char_traits.

Note

> For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the int_type and state_type typedefs, and the eof() member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace std may not be specialized for fundamental types, but classes in namespace __gnu_cxx may be.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html` for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod_char_traits.h.

Definition at line 83 of file char_traits.h.

The documentation for this struct was generated from the following file:

- char_traits.h

4.33 __gnu_cxx::character< V, I, S > Struct Template Reference

Public Types

- typedef character< V, I, S > **char_type**
- typedef I **int_type**
- typedef S **state_type**
- typedef V **value_type**

Static Public Member Functions

- template<typename V2 >
  static char_type **from** (const V2 &v)
- template<typename V2 >
  static V2 **to** (const char_type &c)

Public Attributes

- value_type **value**

4.33.1 Detailed Description

template<typename V, typename I, typename S = std::mbstate_t>struct __gnu_cxx::character< V, I, S >

A POD class that serves as a character abstraction class.

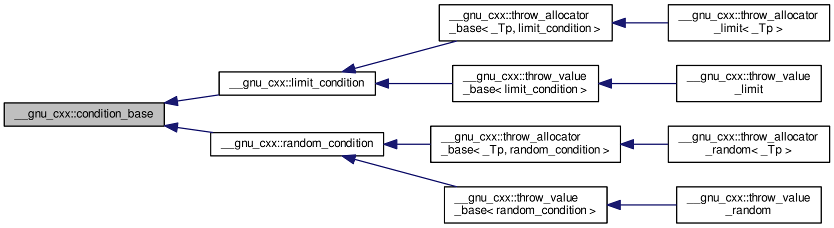
Definition at line 49 of file pod_char_traits.h.

The documentation for this struct was generated from the following file:

- pod_char_traits.h

4.34 __gnu_cxx::condition_base Struct Reference

Inheritance diagram for __gnu_cxx::condition_base:



4.34.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature void throw_conditionally()

Definition at line 247 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.35 __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference

Inherits __gnu_cxx::_Constant_binary_fun< _Result, _Arg1, _Arg2 >.

Public Types

- typedef _Arg1 **first_argument_type**
- typedef _Result **result_type**
- typedef _Arg2 **second_argument_type**

Public Member Functions

- **constant_binary_fun** (const _Result &__v)
- const result_type & **operator()** (const _Arg1 &, const _Arg2 &) const

Public Attributes

- _Result **_M_val**

4.35.1 Detailed Description

template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >

An SGI extension .

Definition at line 320 of file ext/functional.

The documentation for this struct was generated from the following file:

- ext/functional

4.36 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference

Inherits __gnu_cxx::_Constant_unary_fun< _Result, _Argument >.

Public Types

- typedef _Argument **argument_type**
- typedef _Result **result_type**

Public Member Functions

- **constant_unary_fun** (const _Result &__v)
- const result_type & **operator()** (const _Argument &) const

Public Attributes

- result_type **_M_val**

4.36.1 Detailed Description

template<class _Result, class _Argument = _Result>struct __gnu_cxx::constant_unary_fun< _Result, _Argument >

An SGI extension .

Definition at line 312 of file ext/functional.

The documentation for this struct was generated from the following file:

- ext/functional

4.37 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference

Inherits __gnu_cxx::_Constant_void_fun< _Result >.

Public Types

- typedef _Result **result_type**

Public Member Functions

- **constant_void_fun** (const _Result &__v)
- const result_type & **operator()** () const

Public Attributes

- result_type **_M_val**

4.37.1 Detailed Description

template<class _Result>struct __gnu_cxx::constant_void_fun< _Result >

An SGI extension .

Definition at line 303 of file ext/functional.

The documentation for this struct was generated from the following file:

- ext/functional

4.38 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference

Public Types

- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef _Alloc::value_type **value_type**

Public Member Functions

- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const void *__hint)
- void **deallocate** (pointer __p, size_type __n)

4.38.1 Detailed Description

template<typename _Alloc>class __gnu_cxx::debug_allocator< _Alloc >

A meta-allocator with debugging bits, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

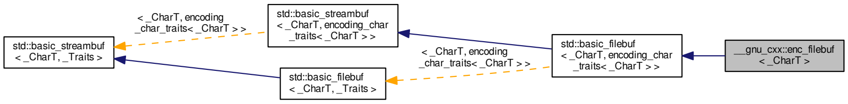
Definition at line 62 of file debug_allocator.h.

The documentation for this class was generated from the following file:

- debug_allocator.h

4.39 __gnu_cxx::enc_filebuf< _CharT > Class Template Reference

Inheritance diagram for __gnu_cxx::enc_filebuf< _CharT >:



Public Types

- typedef codecvt< char_type,
  char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf
  < char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf
  < char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef traits_type::state_type **state_type**
- typedef encoding_char_traits
  < _CharT > **traits_type**

Public Member Functions

- **enc_filebuf** (state_type &__state)
- __filebuf_type ∗ close ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- __filebuf_type ∗ open (const char ∗__s, ios_base::openmode __mode)
- __filebuf_type ∗ open (const std::string &__s, ios_base::openmode __mode)
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type ∗__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)

- int_type sputc (char_type __c)
- streamsize sputn (const char_type ∗__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf ∗ pubsetbuf (char_type ∗__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type ∗, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=encoding_char_traits< _CharT >::eof())
- virtual int_type pbackfail (int_type __c=encoding_char_traits< _CharT >::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type ∗ setbuf (char_type ∗__s, streamsize __n)
- void setg (char_type ∗__gbeg, char_type ∗__gnext, char_type ∗__gend)
- void setp (char_type ∗__pbeg, char_type ∗__pend)
- virtual streamsize showmanyc ()
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type ∗__s, streamsize __n)
- virtual streamsize xsputn (const char_type ∗__s, streamsize __n)

- char_type ∗ eback () const
- char_type ∗ gptr () const
- char_type ∗ egptr () const

- char_type ∗ pbase () const
- char_type ∗ pptr () const
- char_type ∗ epptr () const

Protected Attributes

- char_type ∗ _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type ∗ **_M_codecvt**
- char ∗ _M_ext_buf
- streamsize _M_ext_buf_size
- char ∗ **_M_ext_end**
- const char ∗ _M_ext_next
- __file_type **_M_file**
- char_type ∗ _M_in_beg
- char_type ∗ _M_in_cur
- char_type ∗ _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type ∗ _M_out_beg
- char_type ∗ _M_out_cur
- char_type ∗ _M_out_end
- bool _M_reading
- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type ∗ _M_pback_cur_save
- char_type ∗ _M_pback_end_save
- bool _M_pback_init

4.39.1 Detailed Description

template<typename _CharT>class __gnu_cxx::enc_filebuf< _CharT >

class enc_filebuf.

Definition at line 42 of file enc_filebuf.h.

4.39.2 Member Function Documentation

4.39.2.1 void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_create_pback ( ) [inline], [protected], [inherited]

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back

Definition at line 177 of file fstream.

4.39.2.2 void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_destroy_pback ( ) throw () [inline], [protected], [inherited]

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 194 of file fstream.

4.39.2.3 void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_set_buffer (streamsize *__off*) [inline], [protected], [inherited]

This function sets the pointers of the internal buffer, both get and put areas. Typically:

__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 395 of file fstream.

4.39.2.4 __filebuf_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::close ( ) [inherited]

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

4.39.2.5 char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::eback ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

4.39.2.6 char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::egptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

4.39.2.7 char_type∗ std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::epptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

4.39.2.8 void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gbump (int __n) [inline], [protected], [inherited]

Moving the read position.

Parameters

__n	The delta by which to move.

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

4.39.2.9 locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::getloc ( ) const [inline], [inherited]

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

4.39.2.10 char_type∗ std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

4.39.2.11 virtual void **std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >**::imbue (const locale & *__loc*) `[protected]`, `[virtual]`, `[inherited]`

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.12 streamsize **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >**::in_avail ( ) `[inline]`, `[inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.39.2.13 bool **std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >**::is_open ( ) const throw () `[inline]`, `[inherited]`

Returns true if the external file is open.

Definition at line 227 of file fstream.

4.39.2.14 **__filebuf_type*** **std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >**::open (const char * *__s*, ios_base::openmode *__mode*) `[inherited]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +————————————+ | ios_base Flag combination stdio equivalent | |binary in out trunc app | +————————————+ | + w | | + + a | | + a | | + + w | | + r | | + + r+ | | + + + w+ | | + + + a+ | | + + a+ | +————————————+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + + w+b | | + + + + a+b | | + + + a+b | +————————————+

4.39.2.15 __filebuf_type∗ std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::open (const std::string & __s, ios_base::openmode __mode) [inline], [inherited]

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

Definition at line 280 of file fstream.

4.39.2.16 virtual int_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::overflow (int_type __c = _Traits::eof()) [protected], [virtual], [inherited]

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.17 virtual int_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pbackfail (int_type __c = _Traits::eof()) [protected], [virtual], [inherited]

Tries to back up the input sequence.

Parameters

__c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.18 char_type* **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbase** ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

4.39.2.19 void **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbump** (int *__n*) `[inline]`, `[protected]`, `[inherited]`

Moving the write position.

Parameters

__n	The delta by which to move.

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

4.39.2.20 char_type* **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pptr** ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence

- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

4.39.2.21 locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubimbue (const locale & *__loc*) `[inline]`, `[inherited]`

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

Definition at line 209 of file streambuf.

4.39.2.22 pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekoff (off_type *__off*, ios_base::seekdir *__way*, ios_base::openmode *__mode* = `ios_base::in | ios_base::out`) `[inline]`, `[inherited]`

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.39.2.23 pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekpos (pos_type *__sp*, ios_base::openmode *__mode* = `ios_base::in | ios_base::out`) `[inline]`, `[inherited]`

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

4.39.2.24 basic_streambuf* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubsetbuf (char_type * *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

4.39.2.25 int std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubsync ( ) `[inline]`, `[inherited]`

Calls virtual sync function.

Definition at line 271 of file streambuf.

4.39.2.26 int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sbumpc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file streambuf.

4.39.2.27 virtual pos_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::seekoff (off_type , ios_base::seekdir , ios_base::openmode = `ios_base::in | ios_base::out`) `[protected]`, `[virtual]`, `[inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.28 virtual pos_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::seekpos (pos_type , ios_base::openmode = `ios_base::in | ios_base::out`) `[protected]`, `[virtual]`, `[inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.29 virtual __streambuf_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::setbuf (char_type * *__s*, streamsize *__n*) `[protected]`, `[virtual]`, `[inherited]`

Manipulates the buffer.

Parameters

__s	Pointer to a buffer area.
__n	Size of *__s*.

Returns

`this`

If no file has been opened, and both *__s* and *__n* are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.30 void **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::setg (char_type ∗ *__gbeg*, char_type ∗ *__gnext*, char_type ∗ *__gend*)** `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Definition at line 509 of file streambuf.

4.39.2.31 void **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::setp (char_type ∗ *__pbeg*, char_type ∗ *__pend*)** `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Definition at line 555 of file streambuf.

4.39.2.32 int_type **std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sgetc ( )** `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

4.39.2.33 streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sgetn (char_type * __s, streamsize __n) [inline], [inherited]

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.39.2.34 virtual streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::showmanyc ( ) [protected], [virtual], [inherited]

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.
The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return `eof()` but that they will return immediately.*
The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.35 int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::snextc ( ) [inline], [inherited]

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

4.39.2.36 int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputbackc (char_type __c) [inline], [inherited]

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.

Definition at line 372 of file streambuf.

4.39.2.37 int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputc (char_type __c) [inline], [inherited]

Entry point for all single-character output functions.

Parameters

__c	A character to output.

Returns

__c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *__c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 424 of file streambuf.

4.39.2.38 streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputn (const char_type * __s, streamsize __n) [inline], [inherited]

Entry point for all single-character output functions.

Parameters

__s	A buffer read area.
__n	A count.

One of two public output functions.

Returns xsputn(__s,__n). The effect is to write *__s*[0] through *__s*[__n-1] to the output sequence, if possible.

Definition at line 450 of file streambuf.

4.39.2.39 int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sungetc ( ) `[inline]`, `[inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

4.39.2.40 virtual int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync (void) `[protected]`, `[virtual]`, `[inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.41 virtual int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::uflow ( ) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 700 of file streambuf.

4.39.2.42 virtual int_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::underflow ( ) `[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html`

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.43 virtual streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsgetn (char_type * __s, streamsize __n) `[protected]`, `[virtual]`, `[inherited]`

Multiple character extraction.

Parameters

__s	A buffer area.
__n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.2.44 virtual streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsputn (const char_type * __s, streamsize __n) `[protected]`, `[virtual]`, `[inherited]`

Multiple character insertion.

Parameters

__s	A buffer area.
__n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

4.39.3 Member Data Documentation

4.39.3.1 char_type∗ std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf [protected], [inherited]

Pointer to the beginning of internal buffer.

Definition at line 114 of file fstream.

4.39.3.2 locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_locale [protected], [inherited]

Current locale setting.

Definition at line 192 of file streambuf.

4.39.3.3 size_t std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_size [protected], [inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 121 of file fstream.

4.39.3.4 char∗ std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf [protected], [inherited]

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().

Definition at line 156 of file fstream.

4.39.3.5 streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected], [inherited]

Size of buffer held by _M_ext_buf.

Definition at line 161 of file fstream.

4.39.3.6 const char∗ std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected], [inherited]

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().

Definition at line 168 of file fstream.

4.39.3.7 char_type∗ std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_beg [protected], [inherited]

Start of get area.

Definition at line 184 of file streambuf.

4.39.3.8 char_type∗ std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_cur [protected], [inherited]

Current read area.

Definition at line 185 of file streambuf.

4.39.3.9 char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_end [protected], [inherited]

End of get area.

Definition at line 186 of file streambuf.

4.39.3.10 ios_base::openmode std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_mode [protected], [inherited]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 99 of file fstream.

4.39.3.11 char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_beg [protected], [inherited]

Start of put area.

Definition at line 187 of file streambuf.

4.39.3.12 char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_cur [protected], [inherited]

Current put area.

Definition at line 188 of file streambuf.

4.39.3.13 char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_end [protected], [inherited]

End of put area.

Definition at line 189 of file streambuf.

4.39.3.14 char_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 142 of file fstream.

4.39.3.15 char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_cur_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 143 of file fstream.

4.39.3.16 char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 144 of file fstream.

4.39.3.17 bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init `[protected]`, `[inherited]`

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 145 of file fstream.

4.39.3.18 bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading `[protected]`, `[inherited]`

_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;

NB: _M_reading == true && _M_writing == true is unused.

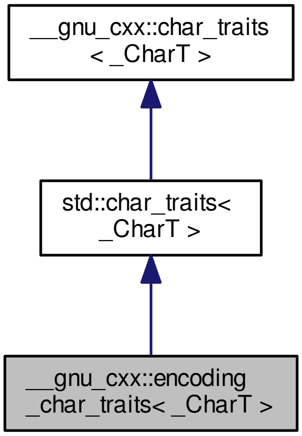
Definition at line 133 of file fstream.

The documentation for this class was generated from the following file:

- enc_filebuf.h

4.40 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference

Inheritance diagram for __gnu_cxx::encoding_char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- typedef _Char_types< _CharT >
  ::int_type **int_type**
- typedef _Char_types< _CharT >
  ::off_type **off_type**
- typedef std::fpos< state_type > **pos_type**
- typedef encoding_state **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type ∗ **assign** (char_type ∗__s, std::size_t __n, char_type __a)
- static int **compare** (const char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static char_type ∗ **copy** (char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type ∗ **find** (const char_type ∗__s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type ∗__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type ∗ **move** (char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

4.40.1 Detailed Description

template<typename _CharT>struct __gnu_cxx::encoding_char_traits< _CharT >

encoding_char_traits

Definition at line 210 of file codecvt_specializations.h.

The documentation for this struct was generated from the following file:

- codecvt_specializations.h

4.41 __gnu_cxx::encoding_state Class Reference

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char ∗__int, const char ∗__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const encoding_state &__obj)
- int **character_ratio** () const
- int **external_bom** () const

- const std::string **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const std::string **internal_encoding** () const
- encoding_state & **operator=** (const encoding_state &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const encoding_state &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- std::string **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- std::string **_M_int_enc**
- descriptor_type **_M_out_desc**

4.41.1 Detailed Description

Extension to use iconv for dealing with character encodings.

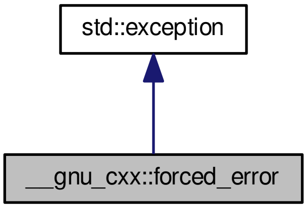
Definition at line 50 of file codecvt_specializations.h.

The documentation for this class was generated from the following file:

- codecvt_specializations.h

4.42 __gnu_cxx::forced_error Struct Reference

Inheritance diagram for __gnu_cxx::forced_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

4.42.1 Detailed Description

Thown by exception safety machinery.

Definition at line 74 of file throw_allocator.h.

4.42.2 Member Function Documentation

4.42.2.1 virtual const char∗ std::exception::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error.

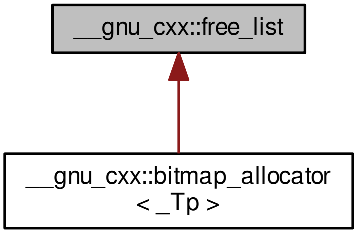
Reimplemented in std::bad_function_call, std::ios_base::failure, std::bad_typeid, std::bad_cast, std::runtime_error, std::future_error, std::bad_exception, std::logic_error, std::bad_weak_ptr, and std::bad_alloc.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.43 __gnu_cxx::free_list Class Reference

Inheritance diagram for __gnu_cxx::free_list:



Public Types

- typedef __mutex **__mutex_type**
- typedef vector_type::iterator **iterator**
- typedef size_t ∗ **value_type**
- typedef __detail::__mini_vector < value_type > **vector_type**

Public Member Functions

- void _M_clear ()
- size_t ∗ _M_get (size_t __sz) throw (std::bad_alloc)
- void _M_insert (size_t ∗__addr) throw ()

4.43.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the bitmap_allocator.

Definition at line 521 of file bitmap_allocator.h.

4.43.2 Member Function Documentation

4.43.2.1 void __gnu_cxx::free_list::_M_clear ( )

This function just clears the internal Free List, and gives back all the memory to the OS.

4.43.2.2 size_t∗ __gnu_cxx::free_list::_M_get (size_t *__sz*) throw (std::bad_alloc)

This function gets a block of memory of the specified size from the free list.

Parameters

__sz	The size in bytes of the memory required.

Returns

A pointer to the new memory block of size at least equal to that requested.

4.43.2.3 void __gnu_cxx::free_list::_M_insert (size_t ∗ *__addr*) throw () [inline]

This function returns the block of memory to the internal free list.

Parameters

__addr	The pointer to the memory block that was given by a call to the _M_get function.

Definition at line 631 of file bitmap_allocator.h.

Referenced by __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object().

The documentation for this class was generated from the following file:

- bitmap_allocator.h

4.44 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Ht::const_pointer **const_pointer**

- typedef _Ht::const_reference **const_reference**
- typedef _Tp **data_type**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Tp **mapped_type**
- typedef _Ht::pointer **pointer**
- typedef _Ht::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- **hash_map** (size_type __n)
- **hash_map** (size_type __n, const hasher &__hf)
- **hash_map** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- template<class _InputIterator >
  hash_map (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator >
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator >
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator >
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __it)
- void **erase** (iterator __f, iterator __l)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- pair< iterator, bool > **insert** (const value_type &__obj)
- template<class _InputIterator >
  void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)

- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- _Tp & **operator[]** (const key_type &__key)
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_map &__hs)

Friends

- template<class _K1 , class _T1 , class _HF , class _EqK , class _Al >
  bool **operator==** (const hash_map< _K1, _T1, _HF, _EqK, _Al > &, const hash_map< _K1, _T1, _HF, _EqK, _Al > &)

4.44.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 83 of file hash_map.

The documentation for this class was generated from the following file:

- hash_map

4.45 __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Ht::const_pointer **const_pointer**
- typedef _Ht::const_reference **const_reference**
- typedef _Tp **data_type**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Tp **mapped_type**
- typedef _Ht::pointer **pointer**
- typedef _Ht::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- **hash_multimap** (size_type __n)
- **hash_multimap** (size_type __n, const hasher &__hf)
- **hash_multimap** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- template< class _InputIterator >
  hash_multimap (_InputIterator __f, _InputIterator __l)
- template< class _InputIterator >
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n)
- template< class _InputIterator >
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template< class _InputIterator >
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __it)
- void **erase** (iterator __f, iterator __l)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- iterator **insert** (const value_type &__obj)
- template< class _InputIterator >
  void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multimap &__hs)

Friends

- template< class _K1 , class _T1 , class _HF , class _EqK , class _Al >
  bool **operator==** (const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &, const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &)

4.45.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 296 of file hash_map.

The documentation for this class was generated from the following file:

- hash_map

4.46 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- **hash_multiset** (size_type __n)
- **hash_multiset** (size_type __n, const hasher &__hf)
- **hash_multiset** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- template<class _InputIterator >
  hash_multiset (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator >
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator >
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator >
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const

- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __it)
- void **erase** (iterator __f, iterator __l)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- iterator **insert** (const value_type &__obj)
- template<class _InputIterator >
  void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multiset &hs)

Friends

- template<class _Val , class _HF , class _EqK , class _Al >
  bool **operator==** (const hash_multiset< _Val, _HF, _EqK, _Al > &, const hash_multiset< _Val, _HF, _EqK, _Al > &)

4.46.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Definition at line 285 of file hash_set.

The documentation for this class was generated from the following file:

- hash_set

4.47 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- **hash_set** (size_type __n)
- **hash_set** (size_type __n, const hasher &__hf)
- **hash_set** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- template<class _InputIterator >
  hash_set (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator >
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator >
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator >
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __it)
- void **erase** (iterator __f, iterator __l)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- pair< iterator, bool > **insert** (const value_type &__obj)
- template<class _InputIterator >
  void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)

- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_set &__hs)

Friends

- template<class _Val , class _HF , class _EqK , class _Al >
  bool **operator==** (const hash_set< _Val, _HF, _EqK, _Al > &, const hash_set< _Val, _HF, _EqK, _Al > &)

4.47.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

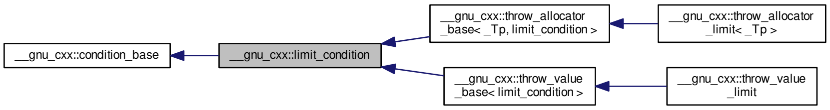
Definition at line 84 of file hash_set.

The documentation for this class was generated from the following file:

- hash_set

4.48 __gnu_cxx::limit_condition Struct Reference

Inheritance diagram for __gnu_cxx::limit_condition:



Classes

- struct always_adjustor

  Always enter the condition.
- struct limit_adjustor

  Enter the nth condition.
- struct never_adjustor

  Never enter the condition.

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

4.48.1 Detailed Description

Base class for incremental control and throw.

Definition at line 256 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.49 __gnu_cxx::limit_condition::always_adjustor Struct Reference

Inherits __gnu_cxx::limit_condition::adjustor_base.

4.49.1 Detailed Description

Always enter the condition.

Definition at line 280 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.50 __gnu_cxx::limit_condition::limit_adjustor Struct Reference

Inherits __gnu_cxx::limit_condition::adjustor_base.

Public Member Functions

- **limit_adjustor** (const size_t __l)

4.50.1 Detailed Description

Enter the nth condition.

Definition at line 286 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.51 __gnu_cxx::limit_condition::never_adjustor Struct Reference

Inherits __gnu_cxx::limit_condition::adjustor_base.

4.51.1 Detailed Description

Never enter the condition.

Definition at line 274 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.52 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference

Public Types

- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **malloc_allocator** (const malloc_allocator &) noexcept
- template<typename _Tp1 >
  malloc_allocator (const malloc_allocator< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void ∗=0)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.52.1 Detailed Description

template<typename _Tp>class __gnu_cxx::malloc_allocator< _Tp >

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

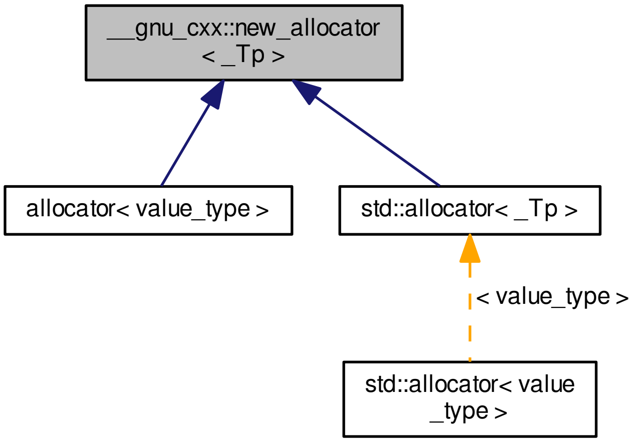
Definition at line 56 of file malloc_allocator.h.

The documentation for this class was generated from the following file:

- malloc_allocator.h

4.53 __gnu_cxx::new_allocator< _Tp > Class Template Reference

Inheritance diagram for __gnu_cxx::new_allocator< _Tp >:



Public Types

- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **new_allocator** (const new_allocator &) noexcept
- template<typename _Tp1 >
  new_allocator (const new_allocator< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void ∗=0)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.53.1 Detailed Description

template<typename _Tp>class __gnu_cxx::new_allocator< _Tp >

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

Template Parameters

_Tp	Type of allocated object.

Definition at line 58 of file new_allocator.h.

The documentation for this class was generated from the following file:

- new_allocator.h

4.54 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference

Inherits __gnu_cxx::_Project1st< _Arg1, _Arg2 >.

Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Arg1 **operator()** (const _Arg1 &__x, const _Arg2 &) const

4.54.1 Detailed Description

template<class _Arg1, class _Arg2>struct __gnu_cxx::project1st< _Arg1, _Arg2 >

An SGI extension .

Definition at line 237 of file ext/functional.

4.54.2 Member Typedef Documentation

4.54.2.1 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.54.2.2 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result **std::binary_function**< _Arg1, _Arg2, _Result >::**result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.54.2.3 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 **std::binary_function**< _Arg1, _Arg2, _Result >::**second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- ext/functional

4.55 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference

Inherits __gnu_cxx::_Project2nd< _Arg1, _Arg2 >.

Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Arg2 **operator()** (const _Arg1 &, const _Arg2 &__y) const

4.55.1 Detailed Description

template<class _Arg1, class _Arg2>struct __gnu_cxx::project2nd< _Arg1, _Arg2 >

An SGI extension .

Definition at line 241 of file ext/functional.

4.55.2 Member Typedef Documentation

4.55.2.1 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 **std::binary_function**< _Arg1, _Arg2, _Result >::**first_argument_type** `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.55.2.2 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result **std::binary_function**< _Arg1, _Arg2, _Result >::**result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.55.2.3 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

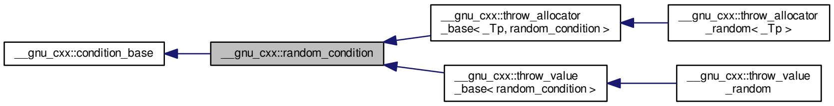
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- ext/functional

4.56 __gnu_cxx::random_condition Struct Reference

Inheritance diagram for __gnu_cxx::random_condition:



Classes

- struct always_adjustor

  Always enter the condition.
- struct group_adjustor

  Group condition.
- struct never_adjustor

  Never enter the condition.

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

4.56.1 Detailed Description

Base class for random probability control and throw.

Definition at line 328 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.57 __gnu_cxx::random_condition::always_adjustor Struct Reference

Inherits __gnu_cxx::random_condition::adjustor_base.

4.57.1 Detailed Description

Always enter the condition.

Definition at line 361 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.58 __gnu_cxx::random_condition::group_adjustor Struct Reference

Inherits __gnu_cxx::random_condition::adjustor_base.

Public Member Functions

- **group_adjustor** (size_t size)

4.58.1 Detailed Description

Group condition.

Definition at line 346 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.59 __gnu_cxx::random_condition::never_adjustor Struct Reference

Inherits __gnu_cxx::random_condition::adjustor_base.

4.59.1 Detailed Description

Never enter the condition.

Definition at line 355 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.60 __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference

Inherits std::_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >.

Public Types

- typedef _Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > **_Base**
- typedef const _Rb_tree_node < _Val > ∗ **_Const_Link_type**
- typedef _Rb_tree_node< _Val > ∗ **_Link_type**
- typedef _Base::allocator_type **allocator_type**
- typedef _Rb_tree_const_iterator < value_type > **const_iterator**
- typedef const value_type ∗ **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Rb_tree_iterator < value_type > **iterator**
- typedef _Key **key_type**
- typedef value_type ∗ **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Val **value_type**

Public Member Functions

- **rb_tree** (const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- bool **__rb_verify** () const
- template<typename... _Args>
  iterator **_M_emplace_equal** (_Args &&...__args)
- template<typename... _Args>
  iterator **_M_emplace_hint_equal** (const_iterator __pos, _Args &&...__args)
- template<typename... _Args>
  iterator **_M_emplace_hint_unique** (const_iterator __pos, _Args &&...__args)
- template<typename... _Args>
  pair< iterator, bool > **_M_emplace_unique** (_Args &&...__args)
- _Node_allocator & **_M_get_Node_allocator** () noexcept
- const _Node_allocator & **_M_get_Node_allocator** () const noexcept
- template<typename _Arg >
  iterator **_M_insert_equal** (_Arg &&__x)
- template<typename _InputIterator >
  void **_M_insert_equal** (_InputIterator __first, _InputIterator __last)
- template<class _II >
  void **_M_insert_equal** (_II __first, _II __last)
- template<typename _Arg >
  iterator **_M_insert_equal_** (const_iterator __position, _Arg &&__x)
- template<typename _Arg >
  pair< iterator, bool > **_M_insert_unique** (_Arg &&__x)
- template<typename _InputIterator >
  void **_M_insert_unique** (_InputIterator __first, _InputIterator __last)

- template<class _II >
  void **_M_insert_unique** (_II __first, _II __last)
- template<typename _Arg >
  iterator **_M_insert_unique_** (const_iterator __position, _Arg &&__x)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- void **clear** () noexcept
- size_type **count** (const key_type &__k) const
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- pair< iterator, iterator > **equal_range** (const key_type &__k)
- pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__k) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- void **erase** (const key_type ∗__first, const key_type ∗__last)
- iterator **find** (const key_type &__k)
- const_iterator **find** (const key_type &__k) const
- allocator_type **get_allocator** () const noexcept
- _Compare **key_comp** () const
- iterator **lower_bound** (const key_type &__k)
- const_iterator **lower_bound** (const key_type &__k) const
- size_type **max_size** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- size_type **size** () const noexcept
- void **swap** (_Rb_tree &__t)
- iterator **upper_bound** (const key_type &__k)
- const_iterator **upper_bound** (const key_type &__k) const

Protected Types

- typedef _Rb_tree_node_base ∗ **_Base_ptr**
- typedef const _Rb_tree_node_base ∗ **_Const_Base_ptr**

Protected Member Functions

- _Link_type **_M_begin** ()
- _Const_Link_type **_M_begin** () const
- _Link_type **_M_clone_node** (_Const_Link_type __x)
- template<typename... _Args>
  _Link_type **_M_create_node** (_Args &&...__args)
- void **_M_destroy_node** (_Link_type __p)
- _Link_type **_M_end** ()
- _Const_Link_type **_M_end** () const

- _Link_type **_M_get_node** ()
- _Base_ptr & **_M_leftmost** ()
- _Const_Base_ptr **_M_leftmost** () const
- void **_M_put_node** (_Link_type __p)
- _Base_ptr & **_M_rightmost** ()
- _Const_Base_ptr **_M_rightmost** () const
- _Base_ptr & **_M_root** ()
- _Const_Base_ptr **_M_root** () const

Static Protected Member Functions

- static const _Key & **_S_key** (_Const_Link_type __x)
- static const _Key & **_S_key** (_Const_Base_ptr __x)
- static _Link_type **_S_left** (_Base_ptr __x)
- static _Const_Link_type **_S_left** (_Const_Base_ptr __x)
- static _Base_ptr **_S_maximum** (_Base_ptr __x)
- static _Const_Base_ptr **_S_maximum** (_Const_Base_ptr __x)
- static _Base_ptr **_S_minimum** (_Base_ptr __x)
- static _Const_Base_ptr **_S_minimum** (_Const_Base_ptr __x)
- static _Link_type **_S_right** (_Base_ptr __x)
- static _Const_Link_type **_S_right** (_Const_Base_ptr __x)
- static const_reference **_S_value** (_Const_Link_type __x)
- static const_reference **_S_value** (_Const_Base_ptr __x)

Protected Attributes

- _Rb_tree_impl< _Compare > **_M_impl**

4.60.1 Detailed Description

template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = allocator<_Value>>struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

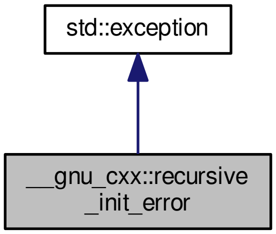
Definition at line 80 of file rb_tree.

The documentation for this struct was generated from the following file:

- rb_tree

4.61 __gnu_cxx::recursive_init_error Class Reference

Inheritance diagram for __gnu_cxx::recursive_init_error:



Public Member Functions

- virtual const char $*$ what () const noexcept

4.61.1 Detailed Description

Exception thrown by __cxa_guard_acquire.

6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 689 of file cxxabi.h.

4.61.2 Member Function Documentation

4.61.2.1 virtual const char$*$ std::exception::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in std::bad_function_call, std::ios_base::failure, std::bad_typeid, std::bad_cast, std::runtime_error, std::future_error, std::bad_exception, std::logic_error, std::bad_weak_ptr, and std::bad_alloc.

The documentation for this class was generated from the following file:

- cxxabi.h

4.62 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference

Inherits __gnu_cxx::_Rope_base< _CharT, _Alloc >.

Public Types

- typedef _Rope_RopeConcatenation< _CharT, _Alloc > **__C**
- typedef _Rope_RopeFunction< _CharT, _Alloc > **__F**
- typedef _Rope_RopeLeaf< _CharT, _Alloc > **__L**
- typedef _Rope_RopeSubstring< _CharT, _Alloc > **__S**
- typedef _Alloc::template rebind< __C >::other **_CAlloc**
- typedef _Alloc::template rebind< _CharT >::other **_DataAlloc**
- typedef _Alloc::template rebind< __F >::other **_FAlloc**
- typedef _Alloc::template rebind< __L >::other **_LAlloc**
- typedef _Alloc::template rebind< __S >::other **_SAlloc**
- typedef _Rope_const_iterator< _CharT, _Alloc > **const_iterator**
- typedef const _CharT ∗ **const_pointer**
- typedef _CharT **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Rope_iterator< _CharT, _Alloc > **iterator**
- typedef _Rope_char_ptr_proxy< _CharT, _Alloc > **pointer**
- typedef _Rope_char_ref_proxy< _CharT, _Alloc > **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _CharT **value_type**

Public Member Functions

- **rope** (const _CharT ∗__s, const allocator_type &__a=allocator_type())
- **rope** (const _CharT ∗__s, size_t __len, const allocator_type &__a=allocator_type())
- **rope** (const _CharT ∗__s, const _CharT ∗__e, const allocator_type &__a=allocator_type())
- **rope** (const const_iterator &__s, const const_iterator &__e, const allocator_type &__a=allocator_type())
- **rope** (const iterator &__s, const iterator &__e, const allocator_type &__a=allocator_type())
- **rope** (_CharT __c, const allocator_type &__a=allocator_type())
- **rope** (size_t __n, _CharT __c, const allocator_type &__a=allocator_type())
- **rope** (const allocator_type &__a=allocator_type())
- **rope** (char_producer< _CharT > ∗__fn, size_t __len, bool __delete_fn, const allocator_type &__a=allocator_type())
- **rope** (const rope &__x, const allocator_type &__a=allocator_type())

- allocator_type & **_M_get_allocator** ()
- const allocator_type & **_M_get_allocator** () const
- rope & **append** (const _CharT ∗__iter, size_t __n)
- rope & **append** (const _CharT ∗__c_string)
- rope & **append** (const _CharT ∗__s, const _CharT ∗__e)
- rope & **append** (const_iterator __s, const_iterator __e)
- rope & **append** (_CharT __c)
- rope & **append** ()
- rope & **append** (const rope &__y)
- rope & **append** (size_t __n, _CharT __c)
- void **apply_to_pieces** (size_t __begin, size_t __end, _Rope_char_consumer< _CharT > &__c) const
- _CharT **at** (size_type __pos) const
- _CharT **back** () const
- void **balance** ()
- const_iterator **begin** () const
- const_iterator **begin** ()
- const _CharT ∗ **c_str** () const
- void **clear** ()
- int **compare** (const rope &__y) const
- const_iterator **const_begin** () const
- const_iterator **const_end** () const
- const_reverse_iterator **const_rbegin** () const
- const_reverse_iterator **const_rend** () const
- void **copy** (_CharT ∗__buffer) const
- size_type **copy** (size_type __pos, size_type __n, _CharT ∗__buffer) const
- void **delete_c_str** ()
- void **dump** ()
- bool **empty** () const
- const_iterator **end** () const
- const_iterator **end** ()
- void **erase** (size_t __p, size_t __n)
- void **erase** (size_t __p)
- iterator **erase** (const iterator &__p, const iterator &__q)
- iterator **erase** (const iterator &__p)
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find** (const _CharT ∗__s, size_type __pos=0) const
- _CharT **front** () const
- allocator_type **get_allocator** () const
- void **insert** (size_t __p, const rope &__r)
- void **insert** (size_t __p, size_t __n, _CharT __c)
- void **insert** (size_t __p, const _CharT ∗__i, size_t __n)
- void **insert** (size_t __p, const _CharT ∗__c_string)
- void **insert** (size_t __p, _CharT __c)
- void **insert** (size_t __p)
- void **insert** (size_t __p, const _CharT ∗__i, const _CharT ∗__j)
- void **insert** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **insert** (size_t __p, const iterator &__i, const iterator &__j)
- iterator **insert** (const iterator &__p, const rope &__r)
- iterator **insert** (const iterator &__p, size_t __n, _CharT __c)
- iterator **insert** (const iterator &__p, _CharT __c)
- iterator **insert** (const iterator &__p)

- iterator **insert** (const iterator &__p, const _CharT ∗c_string)
- iterator **insert** (const iterator &__p, const _CharT ∗__i, size_t __n)
- iterator **insert** (const iterator &__p, const _CharT ∗__i, const _CharT ∗__j)
- iterator **insert** (const iterator &__p, const const_iterator &__i, const const_iterator &__j)
- iterator **insert** (const iterator &__p, const iterator &__i, const iterator &__j)
- size_type **length** () const
- size_type **max_size** () const
- iterator **mutable_begin** ()
- iterator **mutable_end** ()
- reverse_iterator **mutable_rbegin** ()
- reference **mutable_reference_at** (size_type __pos)
- reverse_iterator **mutable_rend** ()
- rope & **operator=** (const rope &__x)
- _CharT **operator[ ]** (size_type __pos) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_CharT __x)
- void **push_front** (_CharT __x)
- const_reverse_iterator **rbegin** () const
- const_reverse_iterator **rbegin** ()
- const_reverse_iterator **rend** () const
- const_reverse_iterator **rend** ()
- void **replace** (size_t __p, size_t __n, const rope &__r)
- void **replace** (size_t __p, size_t __n, const _CharT ∗__i, size_t __i_len)
- void **replace** (size_t __p, size_t __n, _CharT __c)
- void **replace** (size_t __p, size_t __n, const _CharT ∗__c_string)
- void **replace** (size_t __p, size_t __n, const _CharT ∗__i, const _CharT ∗__j)
- void **replace** (size_t __p, size_t __n, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, size_t __n, const iterator &__i, const iterator &__j)
- void **replace** (size_t __p, _CharT __c)
- void **replace** (size_t __p, const rope &__r)
- void **replace** (size_t __p, const _CharT ∗__i, size_t __i_len)
- void **replace** (size_t __p, const _CharT ∗__c_string)
- void **replace** (size_t __p, const _CharT ∗__i, const _CharT ∗__j)
- void **replace** (size_t __p, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_t __p, const iterator &__i, const iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const rope &__r)
- void **replace** (const iterator &__p, const iterator &__q, _CharT __c)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT ∗__c_string)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT ∗__i, size_t __n)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT ∗__i, const _CharT ∗__j)
- void **replace** (const iterator &__p, const iterator &__q, const const_iterator &__i, const const_iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const iterator &__i, const iterator &__j)
- void **replace** (const iterator &__p, const rope &__r)
- void **replace** (const iterator &__p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT ∗__c_string)
- void **replace** (const iterator &__p, const _CharT ∗__i, size_t __n)
- void **replace** (const iterator &__p, const _CharT ∗__i, const _CharT ∗__j)
- void **replace** (const iterator &__p, const_iterator __i, const_iterator __j)
- void **replace** (const iterator &__p, iterator __i, iterator __j)
- const _CharT ∗ **replace_with_c_str** ()

- size_type **size** () const
- rope **substr** (size_t __start, size_t __len=1) const
- rope **substr** (iterator __start, iterator __end) const
- rope **substr** (iterator __start) const
- rope **substr** (const_iterator __start, const_iterator __end) const
- rope< _CharT, _Alloc > **substr** (const_iterator __start)
- void **swap** (rope &__b)

Static Public Member Functions

- static __C ∗ **_C_allocate** (size_t __n)
- static void **_C_deallocate** (__C ∗__p, size_t __n)
- static _CharT ∗ **_Data_allocate** (size_t __n)
- static void **_Data_deallocate** (_CharT ∗__p, size_t __n)
- static __F ∗ **_F_allocate** (size_t __n)
- static void **_F_deallocate** (__F ∗__p, size_t __n)
- static __L ∗ **_L_allocate** (size_t __n)
- static void **_L_deallocate** (__L ∗__p, size_t __n)
- static __S ∗ **_S_allocate** (size_t __n)
- static void **_S_deallocate** (__S ∗__p, size_t __n)

Public Attributes

- _RopeRep ∗ **_M_tree_ptr**

Static Public Attributes

- static const size_type **npos**

Protected Types

- enum { **_S_copy_max** }
- typedef _Rope_base< _CharT, _Alloc > **_Base**
- typedef _CharT ∗ **_Cstrptr**
- typedef _Rope_RopeConcatenation < _CharT, _Alloc > **_RopeConcatenation**
- typedef _Rope_RopeFunction < _CharT, _Alloc > **_RopeFunction**
- typedef _Rope_RopeLeaf< _CharT, _Alloc > **_RopeLeaf**
- typedef _Rope_RopeRep< _CharT, _Alloc > **_RopeRep**
- typedef _Rope_RopeSubstring < _CharT, _Alloc > **_RopeSubstring**
- typedef _Rope_self_destruct_ptr < _CharT, _Alloc > **_Self_destruct_ptr**
- typedef _Base::allocator_type **allocator_type**

Static Protected Member Functions

- static size_t **_S_allocated_capacity** (size_t __n)
- static bool **_S_apply_to_pieces** (_Rope_char_consumer< _CharT > &__c, const _RopeRep ∗__r, size_t __begin, size_t __end)
- static _RopeRep ∗ **_S_concat** (_RopeRep ∗__left, _RopeRep ∗__right)
- static _RopeRep ∗ **_S_concat_char_iter** (_RopeRep ∗__r, const _CharT ∗__iter, size_t __slen)
- static _RopeRep ∗ **_S_destr_concat_char_iter** (_RopeRep ∗__r, const _CharT ∗__iter, size_t __slen)
- static _RopeLeaf ∗ **_S_destr_leaf_concat_char_iter** (_RopeLeaf ∗__r, const _CharT ∗__iter, size_t __slen)
- static _CharT **_S_fetch** (_RopeRep ∗__r, size_type __pos)
- static _CharT ∗ **_S_fetch_ptr** (_RopeRep ∗__r, size_type __pos)
- static bool **_S_is0** (_CharT __c)
- static _RopeLeaf ∗ **_S_leaf_concat_char_iter** (_RopeLeaf ∗__r, const _CharT ∗__iter, size_t __slen)
- static _RopeConcatenation ∗ **_S_new_RopeConcatenation** (_RopeRep ∗__left, _RopeRep ∗__right, allocator_type &__a)
- static _RopeFunction ∗ **_S_new_RopeFunction** (char_producer< _CharT > ∗__f, size_t __size, bool __d, allocator_type &__a)
- static _RopeLeaf ∗ **_S_new_RopeLeaf** (_CharT ∗__s, size_t __size, allocator_type &__a)
- static _RopeSubstring ∗ **_S_new_RopeSubstring** (_Rope_RopeRep< _CharT, _Alloc > ∗__b, size_t __s, size_t __l, allocator_type &__a)
- static void **_S_ref** (_RopeRep ∗__t)
- static _RopeLeaf ∗ **_S_RopeLeaf_from_unowned_char_ptr** (const _CharT ∗__s, size_t __size, allocator_type &__a)
- static size_t **_S_rounded_up_size** (size_t __n)
- static _RopeRep ∗ **_S_substring** (_RopeRep ∗__base, size_t __start, size_t __endp1)
- static _RopeRep ∗ **_S_tree_concat** (_RopeRep ∗__left, _RopeRep ∗__right)
- static void **_S_unref** (_RopeRep ∗__t)
- static _RopeRep ∗ **replace** (_RopeRep ∗__old, size_t __pos1, size_t __pos2, _RopeRep ∗__r)

Static Protected Attributes

- static _CharT **_S_empty_c_str** [1]

Friends

- class **_Rope_char_ptr_proxy< _CharT, _Alloc >**
- class **_Rope_char_ref_proxy< _CharT, _Alloc >**
- class **_Rope_const_iterator< _CharT, _Alloc >**
- class **_Rope_iterator< _CharT, _Alloc >**
- class **_Rope_iterator_base< _CharT, _Alloc >**
- struct **_Rope_RopeRep< _CharT, _Alloc >**
- struct **_Rope_RopeSubstring< _CharT, _Alloc >**
- template<class _CharT2 , class _Alloc2 >
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, const rope< _CharT2, _Alloc2 > &__right)
- template<class _CharT2 , class _Alloc2 >
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, const _CharT2 ∗__right)
- template<class _CharT2 , class _Alloc2 >
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, _CharT2 __right)

4.62.1 Detailed Description

template<class _CharT, class _Alloc>class __gnu_cxx::rope< _CharT, _Alloc >

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

Definition at line 1521 of file rope.

The documentation for this class was generated from the following files:

- rope
- ropeimpl.h

4.63 __gnu_cxx::select1st< _Pair > Struct Template Reference

Inherits std::_Select1st< _Pair >.

Public Types

- typedef _Pair argument_type
- typedef _Pair::first_type result_type

Public Member Functions

- _Pair::first_type & **operator()** (_Pair &__x) const
- const _Pair::first_type & **operator()** (const _Pair &__x) const
- template<typename _Pair2 >
  _Pair2::first_type & **operator()** (_Pair2 &__x) const
- template<typename _Pair2 >
  const _Pair2::first_type & **operator()** (const _Pair2 &__x) const

4.63.1 Detailed Description

template<class _Pair>struct __gnu_cxx::select1st< _Pair >

An SGI extension .

Definition at line 200 of file ext/functional.

4.63.2 Member Typedef Documentation

4.63.2.1 typedef _Pair std::unary_function< _Pair , _Pair::first_type >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.63.2.2 typedef _Pair::first_type **std::unary_function**< _Pair , _Pair::first_type >**::result_type** `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- ext/functional

4.64 __gnu_cxx::select2nd< _Pair > Struct Template Reference

Inherits std::_Select2nd< _Pair >.

Public Types

- typedef _Pair argument_type
- typedef _Pair::second_type result_type

Public Member Functions

- _Pair::second_type & **operator()** (_Pair &__x) const
- const _Pair::second_type & **operator()** (const _Pair &__x) const

4.64.1 Detailed Description

template<class _Pair>struct __gnu_cxx::select2nd< _Pair >

An SGI extension .

Definition at line 205 of file ext/functional.

4.64.2 Member Typedef Documentation

4.64.2.1 typedef _Pair **std::unary_function**< _Pair , _Pair::second_type >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.64.2.2 typedef _Pair::second_type **std::unary_function**< _Pair , _Pair::second_type >**::result_type** `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- ext/functional

4.65 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference

Inherits __gnu_cxx::_Slist_base< _Tp, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Slist_iterator< _Tp,
  const _Tp &, const _Tp ∗ > **const_iterator**
- typedef const value_type ∗ **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Slist_iterator< _Tp,
  _Tp &, _Tp ∗ > **iterator**
- typedef value_type ∗ **pointer**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **slist** (const allocator_type &__a=allocator_type())
- **slist** (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())
- **slist** (size_type __n)
- template<class _InputIterator >
  slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- **slist** (const slist &__x)
- template<class _Integer >
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<class _InputIterator >
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_fill_assign** (size_type __n, const _Tp &__val)
- void **assign** (size_type __n, const _Tp &__val)
- template<class _InputIterator >
  void **assign** (_InputIterator __first, _InputIterator __last)
- iterator **before_begin** ()
- const_iterator **before_begin** () const
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __pos)
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase_after** (iterator __pos)
- iterator **erase_after** (iterator __before_first, iterator __last)
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (iterator __pos, const value_type &__x)
- iterator **insert** (iterator __pos)
- void **insert** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
  void **insert** (iterator __pos, _InIterator __first, _InIterator __last)

- iterator **insert_after** (iterator __pos, const value_type &__x)
- iterator **insert_after** (iterator __pos)
- void **insert_after** (iterator __pos, size_type __n, const value_type &__x)
- template<class _InIterator >
  void **insert_after** (iterator __pos, _InIterator __first, _InIterator __last)
- size_type **max_size** () const
- void **merge** (slist &__x)
- template<class _StrictWeakOrdering >
  void **merge** (slist &, _StrictWeakOrdering)
- slist & **operator=** (const slist &__x)
- void **pop_front** ()
- iterator **previous** (const_iterator __pos)
- const_iterator **previous** (const_iterator __pos) const
- void **push_front** (const value_type &__x)
- void **push_front** ()
- void **remove** (const _Tp &__val)
- template<class _Predicate >
  void **remove_if** (_Predicate __pred)
- void **resize** (size_type new_size, const _Tp &__x)
- void **resize** (size_type new_size)
- void **reverse** ()
- size_type **size** () const
- void **sort** ()
- template<class _StrictWeakOrdering >
  void **sort** (_StrictWeakOrdering __comp)
- void **splice** (iterator __pos, slist &__x)
- void **splice** (iterator __pos, slist &__x, iterator __i)
- void **splice** (iterator __pos, slist &__x, iterator __first, iterator __last)
- void **splice_after** (iterator __pos, iterator __before_first, iterator __before_last)
- void **splice_after** (iterator __pos, iterator __prev)
- void **splice_after** (iterator __pos, slist &__x)
- void **swap** (slist &__x)
- void **unique** ()
- template<class _BinaryPredicate >
  void **unique** (_BinaryPredicate __pred)

Private Types

- typedef _Alloc::template
  rebind< _Slist_node< _Tp >
  >::other **_Node_alloc**

Private Member Functions

- _Slist_node_base ∗ **_M_erase_after** (_Slist_node_base ∗__pos)
- _Slist_node_base ∗ **_M_erase_after** (_Slist_node_base ∗, _Slist_node_base ∗)
- _Slist_node< _Tp > ∗ **_M_get_node** ()
- void **_M_put_node** (_Slist_node< _Tp > ∗__p)

Private Attributes

- _Slist_node_base **_M_head**

4.65.1 Detailed Description

template<class _Tp, class _Alloc = allocator<_Tp>>class __gnu_cxx::slist< _Tp, _Alloc >

This is an SGI extension.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

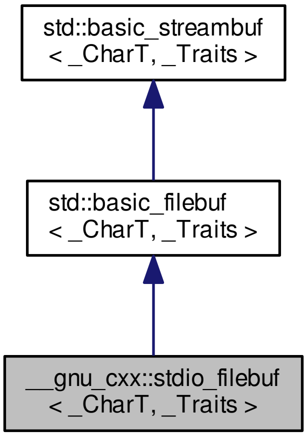
Definition at line 292 of file slist.

The documentation for this class was generated from the following file:

- slist

4.66 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference

Inheritance diagram for __gnu_cxx::stdio_filebuf< _CharT, _Traits >:



Public Types

- typedef codecvt< char_type,
  char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf
  < char_type, traits_type > **__filebuf_type**

- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf < char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef std::size_t **size_t**
- typedef _Traits **traits_type**

Public Member Functions

- stdio_filebuf ()
- stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))
- stdio_filebuf (std::__c_file *__f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))
- virtual ~stdio_filebuf ()
- __filebuf_type * close ()
- int fd ()
- std::__c_file * file ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- __filebuf_type * open (const std::string &__s, ios_base::openmode __mode)
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)

- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=_Traits::eof())
- virtual int_type pbackfail (int_type __c=_Traits::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type ∗ setbuf (char_type ∗__s, streamsize __n)
- void setg (char_type ∗__gbeg, char_type ∗__gnext, char_type ∗__gend)
- void setp (char_type ∗__pbeg, char_type ∗__pend)
- virtual streamsize showmanyc ()
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type ∗__s, streamsize __n)
- virtual streamsize xsputn (const char_type ∗__s, streamsize __n)

- char_type ∗ eback () const
- char_type ∗ gptr () const
- char_type ∗ egptr () const

- char_type ∗ pbase () const
- char_type ∗ pptr () const
- char_type ∗ epptr () const

Protected Attributes

- char_type ∗ _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type ∗ **_M_codecvt**
- char ∗ _M_ext_buf
- streamsize _M_ext_buf_size
- char ∗ **_M_ext_end**
- const char ∗ _M_ext_next
- __file_type **_M_file**
- char_type ∗ _M_in_beg
- char_type ∗ _M_in_cur
- char_type ∗ _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type ∗ _M_out_beg
- char_type ∗ _M_out_cur
- char_type ∗ _M_out_end
- bool _M_reading

- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type ∗ _M_pback_cur_save
- char_type ∗ _M_pback_end_save
- bool _M_pback_init

4.66.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE∗'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

Definition at line 50 of file stdio_filebuf.h.

4.66.2 Constructor & Destructor Documentation

4.66.2.1 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf ( ) `[inline]`

deferred initialization

Definition at line 65 of file stdio_filebuf.h.

4.66.2.2 template<typename _CharT , typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (int *__fd*, std::ios_base::openmode *__mode*, size_t *__size* = `static_cast<size_t>(BUFSIZ)`)

Parameters

__fd	An open file descriptor.
__mode	Same meaning as in a standard filebuf.
__size	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the stdio_filebuf is closed/destroyed.

Definition at line 128 of file stdio_filebuf.h.

4.66.2.3 template<typename _CharT , typename _Traits > __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (std::__c_file ∗ *__f*, std::ios_base::openmode *__mode*, size_t *__size* = `static_cast<size_t>(BUFSIZ)`)

Parameters

__f	An open `FILE∗`.
__mode	Same meaning as in a standard filebuf.
__size	Optimal or preferred size of internal buffer, in chars. Defaults to system's `BUFSIZ`.

This constructor associates a file stream buffer with an open C `FILE∗`. The `FILE∗` will not be automatically closed when the stdio_filebuf is closed/destroyed.

Definition at line 144 of file stdio_filebuf.h.

4.66.2.4 template<typename _CharT , typename _Traits > **__gnu_cxx::stdio_filebuf**< _CharT, _Traits >::~**stdio_filebuf** ( ) `[virtual]`

Closes the external data stream if the file descriptor constructor was used.

Definition at line 123 of file stdio_filebuf.h.

4.66.3 Member Function Documentation

4.66.3.1 template<typename _CharT, typename _Traits> void **std::basic_filebuf**< _CharT, _Traits >::_M_create_pback ( ) `[inline]`, `[protected]`, `[inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back

Definition at line 177 of file fstream.

4.66.3.2 template<typename _CharT, typename _Traits> void **std::basic_filebuf**< _CharT, _Traits >::_M_destroy_pback ( ) throw () `[inline]`, `[protected]`, `[inherited]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 194 of file fstream.

4.66.3.3 template<typename _CharT, typename _Traits> void **std::basic_filebuf**< _CharT, _Traits >::_M_set_buffer (**streamsize** *__off*) `[inline]`, `[protected]`, `[inherited]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 395 of file fstream.

4.66.3.4 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::**__filebuf_type** ∗ **std::basic_filebuf**< _CharT, _Traits >::close ( ) `[inherited]`

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file fstream.tcc.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), and std::basic_filebuf< char_type, traits_type >::~basic_filebuf().

4.66.3.5 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sungetc().

4.66.3.6 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.66.3.7 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.66.3.8 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> int __gnu_cxx::stdio_filebuf< _CharT, _Traits >::fd ( ) [inline]

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 109 of file stdio_filebuf.h.

4.66.3.9 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> std::__c_file* __gnu_cxx::stdio_filebuf< _CharT, _Traits >::file ( ) `[inline]`

Returns

The underlying FILE*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 119 of file stdio_filebuf.h.

4.66.3.10 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::gbump (int __n) `[inline]`, `[protected]`, `[inherited]`

Moving the read position.

Parameters

__n	The delta by which to move.

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.66.3.11 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.66.3.12 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::gptr ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.66.3.13 template<typename _CharT , typename _Traits > void **std::basic_filebuf**< _CharT, _Traits >::imbue (const **locale** & *__loc*) `[protected]`, `[virtual]`, `[inherited]`

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 910 of file fstream.tcc.

References std::ios_base::cur.

4.66.3.14 template<typename _CharT, typename _Traits> streamsize **std::basic_streambuf**< _CharT, _Traits >::in_avail ( ) `[inline]`, `[inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.66.3.15 template<typename _CharT, typename _Traits> bool **std::basic_filebuf**< _CharT, _Traits >::is_open ( ) const throw () `[inline]`, `[inherited]`

Returns true if the external file is open.

Definition at line 227 of file fstream.

Referenced by std::basic_ifstream< _CharT, _Traits >::is_open(), std::basic_ofstream< _CharT, _Traits >::is_open(), and std::basic_fstream< _CharT, _Traits >::is_open().

4.66.3.16 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::**__filebuf_type** ∗ **std::basic_filebuf**< _CharT, _Traits >::open (const char ∗ *__s*, **ios_base::openmode** *__mode*) `[inherited]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +——————————————————+ | ios_base Flag combination stdio equivalent | |binary in out trunc app | +——————————————————+ | + w | | + + a | | + a | | + + w | | + r | | + + r+ | | + + + w+ | | + + + a+ | | + + a+ | +——————————————————————+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + + w+b | | + + + + a+b | | + + + a+b | +————————————————+

Definition at line 94 of file fstream.tcc.

References std::ios_base::ate, std::ios_base::end, and std::basic_filebuf< _CharT, _Traits >::open().

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), and std::basic_fstream< _CharT, _Traits >::open().

4.66.3.17 template<typename _CharT, typename _Traits> **__filebuf_type**∗ **std::basic_filebuf**< _CharT, _Traits >::open (const **std::string** & *__s*, **ios_base::openmode** *__mode*) `[inline]`, `[inherited]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

Definition at line 280 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::open().

4.66.3.18 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::int_type **std::basic_filebuf**< _CharT, _Traits >::overflow (int_type *__c* = `_Traits::eof()`) `[protected]`, `[virtual]`, `[inherited]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 422 of file fstream.tcc.

References std::ios_base::cur, and std::ios_base::out.

4.66.3.19 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (int_type *__c* = `_Traits::eof()`) `[protected], [virtual], [inherited]`

Tries to back up the input sequence.

Parameters

__c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 363 of file fstream.tcc.

References std::ios_base::cur, and std::ios_base::in.

4.66.3.20 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pbase ( ) const `[inline], [protected], [inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.66.3.21 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::pbump (int *__n*) [inline], [protected], [inherited]

Moving the write position.

Parameters

__n	The delta by which to move.

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.66.3.22 template<typename _CharT, typename _Traits> char_type* **std::basic_streambuf**< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.66.3.23 template<typename _CharT, typename _Traits> locale **std::basic_streambuf**< _CharT, _Traits >::pubimbue (const locale & *__loc*) [inline], [inherited]

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

Definition at line 209 of file streambuf.

4.66.3.24 template<typename _CharT, typename _Traits> pos_type **std::basic_streambuf**< _CharT, _Traits >::pubseekoff (**off_type** *__off,* **ios_base::seekdir** *__way,* **ios_base::openmode** *__mode* = **ios_base::in** \| **ios_base::out**) `[inline], [inherited]`

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.66.3.25 template<typename _CharT, typename _Traits> pos_type **std::basic_streambuf**< _CharT, _Traits >::pubseekpos (**pos_type** *__sp,* **ios_base::openmode** *__mode* = **ios_base::in** \| **ios_base::out**) `[inline], [inherited]`

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

4.66.3.26 template<typename _CharT, typename _Traits> **basic_streambuf**∗ **std::basic_streambuf**< _CharT, _Traits >::pubsetbuf (**char_type** ∗ *__s,* **streamsize** *__n*) `[inline], [inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

4.66.3.27 template<typename _CharT, typename _Traits> int **std::basic_streambuf**< _CharT, _Traits >::pubsync ( ) `[inline], [inherited]`

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::sync().

4.66.3.28 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sbumpc ( ) `[inline], [inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::istreambuf_iterator< _CharT, _Traits >::operator++(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.66.3.29 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (off_type , ios_base::seekdir , ios_base::openmode = ios_base::in | ios_base::out) `[protected]`, `[virtual]`, `[inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 713 of file fstream.tcc.

References std::ios_base::cur.

4.66.3.30 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (pos_type , ios_base::openmode = ios_base::in | ios_base::out) `[protected]`, `[virtual]`, `[inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 773 of file fstream.tcc.

References std::ios_base::beg.

4.66.3.31 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >::setbuf (char_type * __s, streamsize __n) `[protected]`, `[virtual]`, `[inherited]`

Manipulates the buffer.

Parameters

__s	Pointer to a buffer area.
__n	Size of *__s*.

Returns

`this`

If no file has been opened, and both *__s* and *__n* are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 684 of file fstream.tcc.

4.66.3.32 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setg (**char_type** * *__gbeg,* **char_type** * *__gnext,* **char_type** * *__gend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Definition at line 509 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.66.3.33 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setp (**char_type** * *__pbeg,* **char_type** * *__pend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Definition at line 555 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.66.3.34 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sgetc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.66.3.35 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s, streamsize __n) `[inline]`, `[inherited]`

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.66.3.36 template<typename _CharT , typename _Traits > streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ( ) `[protected]`, `[virtual]`, `[inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.
The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*
The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 178 of file fstream.tcc.

References std::ios_base::binary, and std::ios_base::in.

4.66.3.37 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.66.3.38 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type __c) `[inline]`, `[inherited]`

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.

Definition at line 372 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::putback().

4.66.3.39 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c) `[inline]`, `[inherited]`

Entry point for all single-character output functions.

Parameters

__c	A character to output.

Returns

__c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *__c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 424 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), and std::ostreambuf_iterator< _CharT, _Traits >::operator=().

4.66.3.40 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * __s, streamsize __n) `[inline]`, `[inherited]`

Entry point for all single-character output functions.

Parameters

__s	A buffer read area.
__n	A count.

One of two public output functions.

Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

Definition at line 450 of file streambuf.

4.66.3.41 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) `[inline]`, `[inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

4.66.3.42 template<typename _CharT , typename _Traits > int std::basic_filebuf< _CharT, _Traits >::sync ( ) `[protected]`, `[virtual]`, `[inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 893 of file fstream.tcc.

4.66.3.43 template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( ) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 700 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc().

4.66.3.44 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ( ) [protected], [virtual], [inherited]

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html`

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 204 of file fstream.tcc.

References std::ios_base::in, and std::min().

4.66.3.45 template<typename _CharT , typename _Traits > streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (char_type * __s, streamsize __n) [protected], [virtual], [inherited]

Multiple character extraction.

Parameters

__s	A buffer area.
__n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills *__s*[0] through *__s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either *__n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 549 of file fstream.tcc.

References std::ios_base::in.

4.66.3.46 template<typename _CharT , typename _Traits > streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (const char_type * __s, streamsize __n) [protected], [virtual], [inherited]

Multiple character insertion.

Parameters

__s	A buffer area.
__n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 637 of file fstream.tcc.

References std::min(), and std::ios_base::out.

4.66.4 Member Data Documentation

4.66.4.1 template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf `[protected]`, `[inherited]`

Pointer to the beginning of internal buffer.

Definition at line 114 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.66.4.2 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale `[protected]`, `[inherited]`

Current locale setting.

Definition at line 192 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf(), std::basic_streambuf< char_type, traits_type >::getloc(), and std::basic_streambuf< char_type, traits_type >::pubimbue().

4.66.4.3 template<typename _CharT, typename _Traits> size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size `[protected]`, `[inherited]`

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 121 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.66.4.4 template<typename _CharT, typename _Traits> char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf `[protected]`, `[inherited]`

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().

Definition at line 156 of file fstream.

4.66.4.5 template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected], [inherited]

Size of buffer held by _M_ext_buf.

Definition at line 161 of file fstream.

4.66.4.6 template<typename _CharT, typename _Traits> const char∗ std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected], [inherited]

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().

Definition at line 168 of file fstream.

4.66.4.7 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]

Start of get area.

Definition at line 184 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::eback(), and std::basic_streambuf< char_type, traits_type >::setg().

4.66.4.8 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]

Current read area.

Definition at line 185 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::gbump(), std::basic_streambuf< char_type, traits_type >::gptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.66.4.9 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]

End of get area.

Definition at line 186 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::egptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.66.4.10 template<typename _CharT, typename _Traits> ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode [protected], [inherited]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 99 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.66.4.11 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]

Start of put area.

Definition at line 187 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbase(), and std::basic_streambuf< char_type, traits_-

type >::setp().

4.66.4.12 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]

Current put area.

Definition at line 188 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbump(), std::basic_streambuf< char_type, traits_type >::pptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.66.4.13 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]

End of put area.

Definition at line 189 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::epptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.66.4.14 template<typename _CharT, typename _Traits> char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 142 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback().

4.66.4.15 template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 143 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), and std::basic_filebuf< char_type, traits_type >::_M_destroy_pback().

4.66.4.16 template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 144 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), and std::basic_filebuf< char_type, traits_type >::_M_destroy_pback().

4.66.4.17 template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 145 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), and std::basic_filebuf< char_type, traits_type >::_M_destroy_pback().

4.66.4.18 template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::_M_reading [protected], [inherited]

_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;

NB: _M_reading == true && _M_writing == true is unused.

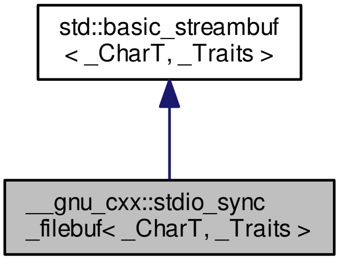
Definition at line 133 of file fstream.

The documentation for this class was generated from the following file:

- stdio_filebuf.h

4.67 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference

Inheritance diagram for __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >:



Public Types

- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**

- typedef _Traits **traits_type**

- typedef basic_streambuf
  < char_type, traits_type > __streambuf_type

Public Member Functions

- **stdio_sync_filebuf** (std::__c_file ∗__f)
- std::__c_file ∗const file ()
- locale getloc () const
- streamsize in_avail ()
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type ∗__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type ∗__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf ∗ pubsetbuf (char_type ∗__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual std::streampos **seekoff** (std::streamoff __off, std::ios_base::seekdir __dir, std::ios_base::openmode=std::ios_base::in|std::ios_base::out)
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual std::streampos **seekpos** (std::streampos __pos, std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf
  < char_type, _Traits > ∗ setbuf (char_type ∗, streamsize)
- void setg (char_type ∗__gbeg, char_type ∗__gnext, char_type ∗__gend)
- void setp (char_type ∗__pbeg, char_type ∗__pend)
- virtual streamsize showmanyc ()
- virtual int sync ()
- int_type **syncgetc** ()

- template<>
  stdio_sync_filebuf< char >
  ::int_type **syncgetc** ()
- template<>
  stdio_sync_filebuf< wchar_t >
  ::int_type **syncgetc** ()
- int_type **syncputc** (int_type __c)
- template<>
  stdio_sync_filebuf< char >
  ::int_type **syncputc** (int_type __c)
- template<>
  stdio_sync_filebuf< wchar_t >
  ::int_type **syncputc** (int_type __c)
- int_type **syncungetc** (int_type __c)
- template<>
  stdio_sync_filebuf< char >
  ::int_type **syncungetc** (int_type __c)
- template<>
  stdio_sync_filebuf< wchar_t >
  ::int_type **syncungetc** (int_type __c)
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual std::streamsize **xsgetn** (char_type ∗__s, std::streamsize __n)
- template<>
  std::streamsize **xsgetn** (char ∗__s, std::streamsize __n)
- template<>
  std::streamsize **xsgetn** (wchar_t ∗__s, std::streamsize __n)
- virtual streamsize xsgetn (char_type ∗__s, streamsize __n)
- virtual std::streamsize **xsputn** (const char_type ∗__s, std::streamsize __n)
- template<>
  std::streamsize **xsputn** (const char ∗__s, std::streamsize __n)
- template<>
  std::streamsize **xsputn** (const wchar_t ∗__s, std::streamsize __n)
- virtual streamsize xsputn (const char_type ∗__s, streamsize __n)

- char_type ∗ eback () const
- char_type ∗ gptr () const
- char_type ∗ egptr () const

- char_type ∗ pbase () const
- char_type ∗ pptr () const
- char_type ∗ epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type ∗ _M_in_beg
- char_type ∗ _M_in_cur
- char_type ∗ _M_in_end
- char_type ∗ _M_out_beg
- char_type ∗ _M_out_cur
- char_type ∗ _M_out_end

4.67.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE∗'s. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

Definition at line 56 of file stdio_sync_filebuf.h.

4.67.2 Member Typedef Documentation

4.67.2.1 template<typename _CharT, typename _Traits> typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__streambuf_type [inherited]

This is a non-standard type.

Definition at line 138 of file streambuf.

4.67.3 Member Function Documentation

4.67.3.1 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sungetc().

4.67.3.2 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.67.3.3 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.67.3.4 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> std::__c_file* const __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file ( ) [inline]

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 88 of file stdio_sync_filebuf.h.

4.67.3.5 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::gbump (int __n) [inline], [protected], [inherited]

Moving the read position.

Parameters

__n	The delta by which to move.

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.67.3.6 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.67.3.7 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.67.3.8 template<typename _CharT, typename _Traits> virtual void std::basic_streambuf< _CharT, _Traits >::imbue (const locale & __loc) [inline], [protected], [virtual], [inherited]

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 576 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.67.3.9 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( ) [inline], [inherited]

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.67.3.10 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> virtual **int_type** __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::overflow (**int_type** *__c* = `traits_type::eof()`) `[inline]`, `[protected]`, `[virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 141 of file stdio_sync_filebuf.h.

4.67.3.11 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> virtual **int_type** __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pbackfail (**int_type** *__c* = `traits_type::eof()`) `[inline]`, `[protected]`, `[virtual]`

Tries to back up the input sequence.

Parameters

__c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 116 of file stdio_sync_filebuf.h.

4.67.3.12 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::pbase ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.67.3.13 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::pbump (int __n) [inline], [protected], [inherited]

Moving the write position.

Parameters

__n	The delta by which to move.

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.67.3.14 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.67.3.15 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale & __loc) [inline], [inherited]

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

Definition at line 209 of file streambuf.

4.67.3.16 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type *__off,* ios_base::seekdir *__way,* ios_base::openmode *__mode =* ios_base::in | ios_base::out) `[inline]`, `[inherited]`

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.67.3.17 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type *__sp,* ios_base::openmode *__mode =* ios_base::in | ios_base::out) `[inline]`, `[inherited]`

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

4.67.3.18 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (char_type * *__s,* streamsize *__n*) `[inline]`, `[inherited]`

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

4.67.3.19 template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) `[inline]`, `[inherited]`

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::sync().

4.67.3.20 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::istreambuf_iterator< _CharT, _Traits >::operator++(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.67.3.21 template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (off_type, ios_base::seekdir, ios_base::openmode = ios_base::in | ios_base::out) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 602 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubseekoff().

4.67.3.22 template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (pos_type, ios_base::openmode = ios_base::in | ios_base::out) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 614 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubseekpos().

4.67.3.23 template<typename _CharT, typename _Traits> virtual basic_streambuf<char_type,_Traits>* std::basic_streambuf< _CharT, _Traits >::setbuf (char_type *, streamsize) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 591 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubsetbuf().

4.67.3.24 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setg (char_type ∗ *__gbeg*, char_type ∗ *__gnext*, char_type ∗ *__gend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Definition at line 509 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.67.3.25 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setp (char_type ∗ *__pbeg*, char_type ∗ *__pend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Definition at line 555 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.67.3.26 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.67.3.27 template<typename _CharT, typename _Traits> streamsize **std::basic_streambuf**< _CharT, _Traits >::sgetn (**char_type** ∗ ***__s*, streamsize *__n***) `[inline]`, `[inherited]`

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.67.3.28 template<typename _CharT, typename _Traits> virtual streamsize **std::basic_streambuf**< _CharT, _Traits >::showmanyc ( ) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.
The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*
The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 649 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::in_avail().

4.67.3.29 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::snextc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.67.3.30 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type __c) `[inline]`, `[inherited]`

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.

Definition at line 372 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::putback().

4.67.3.31 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c) `[inline]`, `[inherited]`

Entry point for all single-character output functions.

Parameters

__c	A character to output.

Returns

__c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *__c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 424 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), and std::ostreambuf_iterator< _CharT, _Traits >::operator=().

4.67.3.32 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * __s, streamsize __n) `[inline]`, `[inherited]`

Entry point for all single-character output functions.

Parameters

__s	A buffer read area.
__n	A count.

One of two public output functions.

Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

Definition at line 450 of file streambuf.

4.67.3.33 template<typename _CharT, typename _Traits> **int_type std::basic_streambuf**< _CharT, _Traits >::sungetc ( ) `[inline]`, `[inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

4.67.3.34 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> virtual int **__gnu_cxx::stdio_sync_filebuf**< _CharT, _Traits >::sync (void) `[inline]`, `[protected]`, `[virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 160 of file stdio_sync_filebuf.h.

4.67.3.35 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> virtual **int_type __gnu_cxx::stdio_sync_filebuf**< _CharT, _Traits >::uflow ( ) `[inline]`, `[protected]`, `[virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 108 of file stdio_sync_filebuf.h.

4.67.3.36 template<typename _CharT , typename _Traits = std::char_traits<_CharT>> virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::underflow ( ) [inline], [protected], [virtual]

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html`

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 101 of file stdio_sync_filebuf.h.

4.67.3.37 template<typename _CharT , typename _Traits > streamsize basic_streambuf< _CharT, _Traits >::xsgetn (char_type ∗ __s, streamsize __n) [protected], [virtual], [inherited]

Multiple character extraction.

Parameters

__s	A buffer area.
__n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 46 of file streambuf.tcc.

References std::min().

Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

4.67.3.38 template<typename _CharT , typename _Traits > streamsize basic_streambuf< _CharT, _Traits >::xsputn (const char_type ∗ __s, streamsize __n) [protected], [virtual], [inherited]

Multiple character insertion.

Parameters

__s	A buffer area.
__n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 80 of file streambuf.tcc.

References std::min().

Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

4.67.4 Member Data Documentation

4.67.4.1 template<typename _CharT, typename _Traits> locale **std::basic_streambuf**< _CharT, _Traits >::_M_buf_locale `[protected]`, `[inherited]`

Current locale setting.

Definition at line 192 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf(), std::basic_streambuf< char_type, traits_type >::getloc(), and std::basic_streambuf< char_type, traits_type >::pubimbue().

4.67.4.2 template<typename _CharT, typename _Traits> **char_type*** **std::basic_streambuf**< _CharT, _Traits >::_M_in_beg `[protected]`, `[inherited]`

Start of get area.

Definition at line 184 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::eback(), and std::basic_streambuf< char_type, traits_type >::setg().

4.67.4.3 template<typename _CharT, typename _Traits> **char_type*** **std::basic_streambuf**< _CharT, _Traits >::_M_in_cur `[protected]`, `[inherited]`

Current read area.

Definition at line 185 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::gbump(), std::basic_streambuf< char_type, traits_type >::gptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.67.4.4 template<typename _CharT, typename _Traits> **char_type*** **std::basic_streambuf**< _CharT, _Traits >::_M_in_end `[protected]`, `[inherited]`

End of get area.

Definition at line 186 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::egptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.67.4.5 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]

Start of put area.

Definition at line 187 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbase(), and std::basic_streambuf< char_type, traits_type >::setp().

4.67.4.6 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]

Current put area.

Definition at line 188 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbump(), std::basic_streambuf< char_type, traits_type >::pptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.67.4.7 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]

End of put area.

Definition at line 189 of file streambuf.

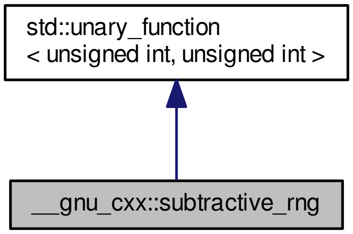
Referenced by std::basic_streambuf< char_type, traits_type >::epptr(), and std::basic_streambuf< char_type, traits_type >::setp().

The documentation for this class was generated from the following file:

- stdio_sync_filebuf.h

4.68 __gnu_cxx::subtractive_rng Class Reference

Inheritance diagram for __gnu_cxx::subtractive_rng:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

Public Member Functions

- subtractive_rng (unsigned int __seed)
- subtractive_rng ()
- void **_M_initialize** (unsigned int __seed)
- unsigned int operator() (unsigned int __limit)

4.68.1 Detailed Description

The `subtractive_rng` class is documented on `SGI's site`. Note that this code assumes that `int` is 32 bits.

Definition at line 352 of file ext/functional.

4.68.2 Member Typedef Documentation

4.68.2.1 template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.68.2.2 template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

4.68.3 Constructor & Destructor Documentation

4.68.3.1 __gnu_cxx::subtractive_rng::subtractive_rng (unsigned int *__seed*) `[inline]`

Ctor allowing you to initialize the seed.

Definition at line 394 of file ext/functional.

4.68.3.2 __gnu_cxx::subtractive_rng::subtractive_rng ( ) `[inline]`

Default ctor; initializes its state with some number you don't see.

Definition at line 398 of file ext/functional.

4.68.4 Member Function Documentation

4.68.4.1 unsigned int __gnu_cxx::subtractive_rng::operator() (unsigned int *__limit*) `[inline]`

Returns a number less than the argument.

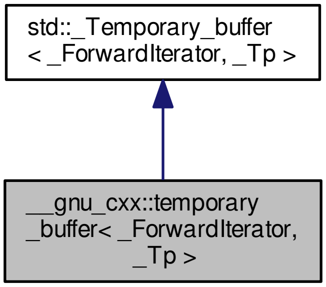
Definition at line 363 of file ext/functional.

The documentation for this class was generated from the following file:

- ext/functional

4.69 __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp > Struct Template Reference

Inheritance diagram for __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type ∗ **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- temporary_buffer (_ForwardIterator __first, _ForwardIterator __last)
- ∼temporary_buffer ()
- iterator begin ()
- iterator end ()
- size_type requested_size () const
- size_type size () const

Protected Attributes

- pointer **_M_buffer**
- size_type **_M_len**
- size_type **_M_original_len**

4.69.1 Detailed Description

template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >

This class provides similar behavior and semantics of the standard functions get_temporary_buffer() and return_temporary_buffer(), but encapsulated in a type vaguely resembling a standard container.

By default, a temporary_buffer<Iter> stores space for objects of whatever type the Iter iterator points to. It is constructed from a typical [first,last) range, and provides the begin(), end(), size() functions, as well as requested_size(). For non-trivial types, copies of ∗first will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like get_temporary_buffer(), not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of [first,last), but if size() is less than requested_size(), then this didn't happen.

Definition at line 183 of file ext/memory.

4.69.2 Constructor & Destructor Documentation

4.69.2.1 template<class _ForwardIterator , class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type> __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer (_ForwardIterator *__first*, _ForwardIterator *__last*) `[inline]`

Requests storage large enough to hold a copy of [first,last).

Definition at line 186 of file ext/memory.

4.69.2.2 template<class _ForwardIterator , class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type> __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::∼temporary_buffer ( ) `[inline]`

Destroys objects and frees storage.

Definition at line 190 of file ext/memory.

4.69.3 Member Function Documentation

4.69.3.1 template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( ) `[inline]`, `[inherited]`

As per Table mumble.

Definition at line 151 of file stl_tempbuf.h.

Referenced by std::inplace_merge(), std::stable_partition(), and std::stable_sort().

4.69.3.2 template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( ) `[inline]`, `[inherited]`

As per Table mumble.

Definition at line 156 of file stl_tempbuf.h.

4.69.3.3 template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size ( ) const `[inline]`, `[inherited]`

Returns the size requested by the constructor; may be >size().

Definition at line 146 of file stl_tempbuf.h.

Referenced by std::stable_partition().

4.69.3.4 template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const [inline], [inherited]

As per Table mumble.

Definition at line 141 of file stl_tempbuf.h.

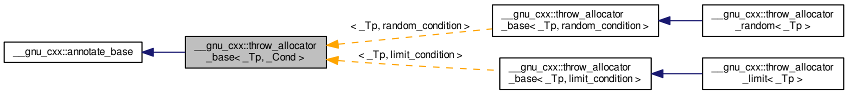
Referenced by std::inplace_merge(), std::stable_partition(), and std::stable_sort().

The documentation for this struct was generated from the following file:

- ext/memory

4.70 __gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference

Inheritance diagram for __gnu_cxx::throw_allocator_base< _Tp, _Cond >:



Public Types

- typedef const value_type ∗ **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, std::allocator< void >::const_pointer hint=0)
- void **check_allocated** (void ∗p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (size_type __n)
- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up >
  void **destroy** (_Up ∗__p)

- void **erase** (void ∗p, size_t size)
- void **insert** (void ∗p, size_t size)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)

4.70.1 Detailed Description

template<typename _Tp, typename _Cond>class __gnu_cxx::throw_allocator_base< _Tp, _Cond >

Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templatized code.

Note: Deallocate not allowed to throw.

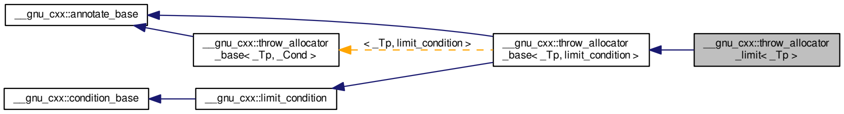
Definition at line 634 of file throw_allocator.h.

The documentation for this class was generated from the following file:

- throw_allocator.h

4.71 __gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference

Inheritance diagram for __gnu_cxx::throw_allocator_limit< _Tp >:



Public Types

- typedef const value_type ∗ **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_limit** (const throw_allocator_limit &) noexcept
- template<typename _Tp1 >
  throw_allocator_limit (const throw_allocator_limit< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, std::allocator< void >::const_pointer hint=0)
- void **check_allocated** (void ∗p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (size_type __n)
- void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up ∗__p)
- void **erase** (void ∗p, size_t size)
- void **insert** (void ∗p, size_t size)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

4.71.1 Detailed Description

template<typename _Tp>struct __gnu_cxx::throw_allocator_limit< _Tp >

Allocator throwing via limit condition.

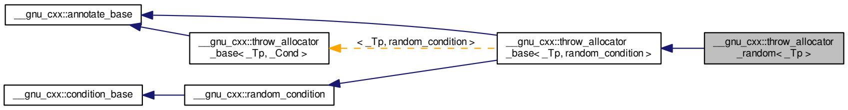
Definition at line 737 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.72 __gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference

Inheritance diagram for __gnu_cxx::throw_allocator_random< _Tp >:



Public Types

- typedef const value_type ∗ **const_pointer**
- typedef const value_type & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef value_type ∗ **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_random** (const throw_allocator_random &) noexcept
- template<typename _Tp1 >
  throw_allocator_random (const throw_allocator_random< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, std::allocator< void >::const_pointer hint=0)
- void **check_allocated** (void ∗p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_allocated** (size_type __n)
- void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up ∗__p)
- void **erase** (void ∗p, size_t size)
- void **insert** (void ∗p, size_t size)
- size_type **max_size** () const noexcept
- void **seed** (unsigned long __s)

Static Public Member Functions

- static size_t **get_label** ()
- static void **set_label** (size_t l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

4.72.1 Detailed Description

template<typename _Tp>struct __gnu_cxx::throw_allocator_random< _Tp >

Allocator throwing via random condition.

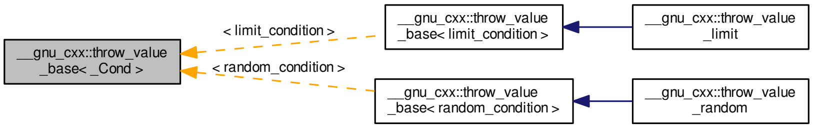
Definition at line 758 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.73 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference

Inheritance diagram for __gnu_cxx::throw_value_base< _Cond >:



Public Types

- typedef _Cond **condition_type**

Public Member Functions

- **throw_value_base** (const throw_value_base &__v)
- **throw_value_base** (throw_value_base &&)=default
- **throw_value_base** (const std::size_t __i)
- throw_value_base & **operator++** ()
- throw_value_base & **operator=** (const throw_value_base &__v)
- throw_value_base & **operator=** (throw_value_base &&)=default

Public Attributes

- std::size_t **_M_i**

4.73.1 Detailed Description

template<typename _Cond>struct __gnu_cxx::throw_value_base< _Cond >

Class with exception generation control. Intended to be used as a value_type in templatized code.

Note: Destructor not allowed to throw.

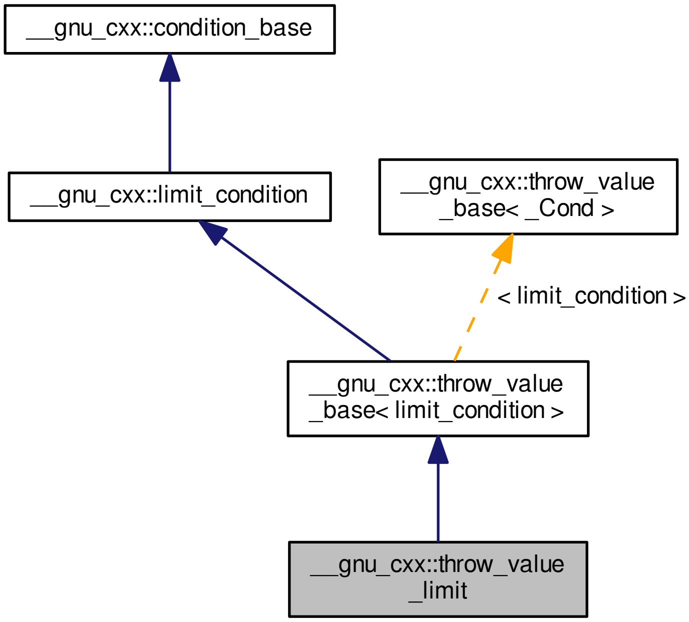
Definition at line 447 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.74 __gnu_cxx::throw_value_limit Struct Reference

Inheritance diagram for __gnu_cxx::throw_value_limit:



Public Types

- typedef throw_value_base < limit_condition > **base_type**
- typedef limit_condition **condition_type**

Public Member Functions

- **throw_value_limit** (const throw_value_limit &__other)
- **throw_value_limit** (throw_value_limit &&)=default
- **throw_value_limit** (const std::size_t __i)
- throw_value_base & **operator++** ()
- throw_value_limit & **operator=** (const throw_value_limit &__other)
- throw_value_limit & **operator=** (throw_value_limit &&)=default

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)

- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**

4.74.1 Detailed Description

Type throwing via limit condition.

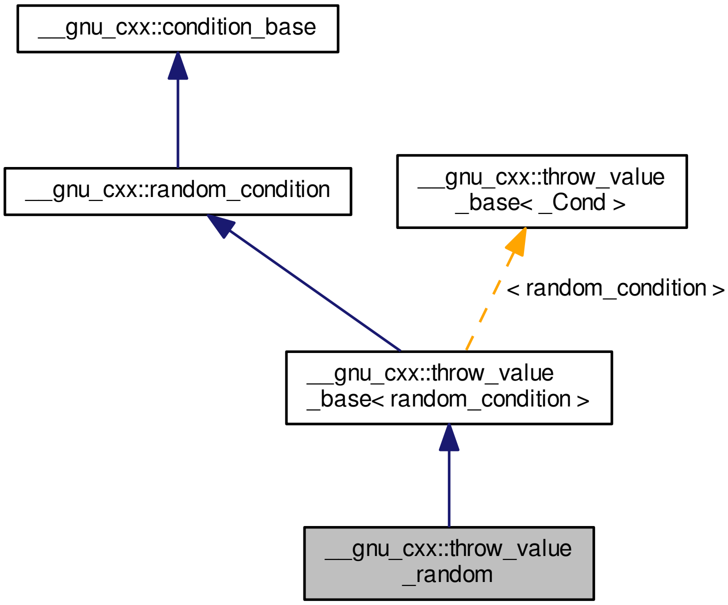
Definition at line 564 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.75 __gnu_cxx::throw_value_random Struct Reference

Inheritance diagram for __gnu_cxx::throw_value_random:



Public Types

- typedef throw_value_base < random_condition > **base_type**
- typedef random_condition **condition_type**

Public Member Functions

- **throw_value_random** (const throw_value_random &__other)
- **throw_value_random** (throw_value_random &&)=default
- **throw_value_random** (const std::size_t __i)
- throw_value_base & **operator++** ()
- throw_value_random & **operator=** (const throw_value_random &__other)
- throw_value_random & **operator=** (throw_value_random &&)=default
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**

4.75.1 Detailed Description

Type throwing via random condition.

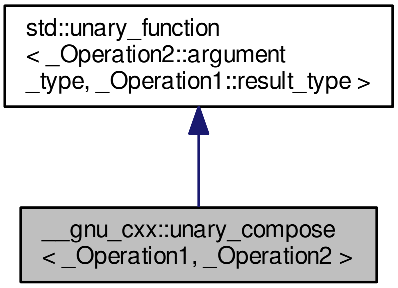
Definition at line 595 of file throw_allocator.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.76 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference

Inheritance diagram for __gnu_cxx::unary_compose< _Operation1, _Operation2 >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

Public Member Functions

- **unary_compose** (const _Operation1 &__x, const _Operation2 &__y)
- _Operation1::result_type **operator()** (const typename _Operation2::argument_type &__x) const

Protected Attributes

- _Operation1 **_M_fn1**
- _Operation2 **_M_fn2**

4.76.1 Detailed Description

template<class _Operation1, class _Operation2>class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

An SGI extension .

Definition at line 125 of file ext/functional.

4.76.2 Member Typedef Documentation

4.76.2.1 template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.76.2.2 template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- ext/functional

4.77 __gnu_debug::_After_nth_from< _Iterator > Class Template Reference

Public Member Functions

- **_After_nth_from** (const difference_type &__n, const _Iterator &__base)
- bool **operator()** (const _Iterator &__x) const

4.77.1 Detailed Description

template<typename _Iterator>class __gnu_debug::_After_nth_from< _Iterator >

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 77 of file safe_sequence.h.

The documentation for this class was generated from the following file:

- safe_sequence.h

4.78 __gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference

Public Types

- typedef _It::iterator_type **_BaseIt**
- typedef _Sequence::const_iterator **_It**

Static Public Member Functions

- static bool **_S_Is** (_BaseIt, const _Sequence ∗)
- static bool **_S_Is_Beginnest** (_BaseIt __it, const _Sequence ∗__seq)

4.78.1 Detailed Description

template<typename _Sequence>struct __gnu_debug::_BeforeBeginHelper< _Sequence >

Helper struct to deal with sequence offering a before_begin iterator.

Definition at line 45 of file safe_iterator.h.

The documentation for this struct was generated from the following file:

- safe_iterator.h

4.79 __gnu_debug::_Equal_to< _Type > Class Template Reference

Public Member Functions

- **_Equal_to** (const _Type &__v)
- bool **operator()** (const _Type &__x) const

4.79.1 Detailed Description

template<typename _Type>class __gnu_debug::_Equal_to< _Type >

A simple function object that returns true if the passed-in value is equal to the stored value.

Definition at line 62 of file safe_sequence.h.

The documentation for this class was generated from the following file:

- safe_sequence.h

4.80 __gnu_debug::_Not_equal_to< _Type > Class Template Reference

Public Member Functions

- **_Not_equal_to** (const _Type &__v)
- bool **operator()** (const _Type &__x) const

4.80.1 Detailed Description

template<typename _Type>class __gnu_debug::_Not_equal_to< _Type >

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both bind1st and not_equal.

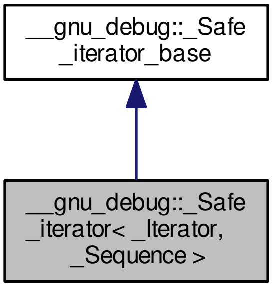
Definition at line 47 of file safe_sequence.h.

The documentation for this class was generated from the following file:

- safe_sequence.h

4.81 __gnu_debug::_Safe_iterator< _Iterator, _Sequence > Class Template Reference

Inheritance diagram for __gnu_debug::_Safe_iterator< _Iterator, _Sequence >:



Public Types

- typedef _Traits::difference_type **difference_type**
- typedef _Traits::iterator_category **iterator_category**
- typedef _Iterator **iterator_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::value_type **value_type**

Public Member Functions

- _Safe_iterator ()
- _Safe_iterator (const _Iterator &__i, const _Sequence ∗__seq)
- _Safe_iterator (const _Safe_iterator &__x)
- _Safe_iterator (_Safe_iterator &&__x)
- template<typename _MutableIterator >
  _Safe_iterator (const _Safe_iterator< _MutableIterator, typename __gnu_cxx::__enable_if<(std::__are_same< _MutableIterator, typename _Sequence::iterator::iterator_type >::__value), _Sequence >::__type > &__x)
- void _M_attach (_Safe_sequence_base ∗__seq, bool __constant)
- void _M_attach (_Safe_sequence_base ∗__seq)
- void _M_attach_single (_Safe_sequence_base ∗__seq, bool __constant) throw ()
- void _M_attach_single (_Safe_sequence_base ∗__seq)
- bool _M_attached_to (const _Safe_sequence_base ∗__seq) const
- bool _M_before_dereferenceable () const
- bool **_M_can_advance** (const difference_type &__n) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const throw ()
- bool **_M_decrementable** () const
- bool _M_dereferenceable () const
- void _M_detach ()
- void _M_detach_single () throw ()
- const _Sequence ∗ **_M_get_sequence** () const
- bool _M_incrementable () const
- void _M_invalidate ()
- bool _M_is_before_begin () const
- bool _M_is_begin () const
- bool _M_is_beginnest () const
- bool _M_is_end () const
- void _M_reset () throw ()
- bool _M_singular () const throw ()
- void _M_unlink () throw ()
- template<typename _Other >
  bool **_M_valid_range** (const _Safe_iterator< _Other, _Sequence > &__rhs) const
- _Iterator base () const
- operator _Iterator () const
- reference operator∗ () const
- _Safe_iterator **operator+** (const difference_type &__n) const
- _Safe_iterator & operator++ ()
- _Safe_iterator operator++ (int)
- _Safe_iterator & **operator+=** (const difference_type &__n)
- _Safe_iterator **operator-** (const difference_type &__n) const
- _Safe_iterator & operator-- ()
- _Safe_iterator operator-- (int)
- _Safe_iterator & **operator-=** (const difference_type &__n)
- pointer operator-> () const
- _Safe_iterator & operator= (const _Safe_iterator &__x)
- _Safe_iterator & operator= (_Safe_iterator &&__x)
- reference **operator[ ]** (const difference_type &__n) const

Public Attributes

- _Safe_iterator_base ∗ _M_next
- _Safe_iterator_base ∗ _M_prior
- _Safe_sequence_base ∗ _M_sequence
- unsigned int _M_version

Protected Member Functions

- __gnu_cxx::__mutex & _M_get_mutex () throw ()

4.81.1 Detailed Description

template<typename _Iterator, typename _Sequence>class __gnu_debug::_Safe_iterator< _Iterator, _Sequence >

Safe iterator wrapper.

The class template _Safe_iterator is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In additional to the basic iterator operations (which are validated, and then passed to the underlying iterator), _Safe_iterator has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 116 of file safe_iterator.h.

4.81.2 Constructor & Destructor Documentation

4.81.2.1 template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator ( ) `[inline]`

Postcondition

the iterator is singular and unattached

Definition at line 142 of file safe_iterator.h.

4.81.2.2 template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator (const _Iterator & *__i*, const _Sequence ∗ *__seq*) `[inline]`

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 151 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_singular().

4.81.2.3 template<typename _Iterator, typename _Sequence> **__gnu_debug::_Safe_iterator**< _Iterator, _Sequence >::**_Safe_iterator** (const **_Safe_iterator**< _Iterator, _Sequence > & *__x*) `[inline]`

Copy construction.

Definition at line 162 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_singular().

4.81.2.4 template<typename _Iterator, typename _Sequence> **__gnu_debug::_Safe_iterator**< _Iterator, _Sequence >::**_Safe_iterator** (**_Safe_iterator**< _Iterator, _Sequence > && *__x*) `[inline]`

Move construction.

Postcondition

__x is singular and unattached

Definition at line 179 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach().

4.81.2.5 template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > **__gnu_debug::_Safe_iterator**< _Iterator, _Sequence >::**_Safe_iterator** (const **_Safe_iterator**< _MutableIterator, typename __gnu_cxx::__enable_if<(std::__are_same< _MutableIterator, typename _Sequence::iterator::iterator_type >::__value), _Sequence >::__type > & *__x*) `[inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 197 of file safe_iterator.h.

4.81.3 Member Function Documentation

4.81.3.1 void __gnu_debug::_Safe_iterator_base::_M_attach (**_Safe_sequence_base** ∗ *__seq,* bool *__constant*) `[inherited]`

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach(), and __gnu_debug::_Safe_iterator_base::_Safe_iterator_base().

4.81.3.2 template<typename _Iterator, typename _Sequence> void **__gnu_debug::_Safe_iterator**< _Iterator, _Sequence >::_M_attach (**_Safe_sequence_base** ∗ *__seq*) `[inline]`

Attach iterator to the given sequence.

Definition at line 406 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_attach().

Referenced by __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.81.3.3 void __gnu_debug::_Safe_iterator_base::_M_attach_single (**_Safe_sequence_base** ∗ *__seq,* bool *__constant*) throw () `[inherited]`

Likewise, but not thread-safe.

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach_single(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach_single().

4.81.3.4 template<typename _Iterator, typename _Sequence> void __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach_single (_Safe_sequence_base ∗ __seq) [inline]

Likewise, but not thread-safe.

Definition at line 413 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_attach_single().

4.81.3.5 bool __gnu_debug::_Safe_iterator_base::_M_attached_to (const _Safe_sequence_base ∗ __seq) const [inline], [inherited]

Determines if we are attached to the given sequence.

Definition at line 129 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_sequence.

4.81.3.6 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_before_dereferenceable ( ) const [inline]

Is the iterator before a dereferenceable one?

Definition at line 425 of file safe_iterator.h.

References __gnu_debug::__base(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base().

4.81.3.7 bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw () [inherited]

Can we compare this iterator to the given iterator ___x? Returns true if both iterators are nonsingular and reference the same sequence.

4.81.3.8 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable ( ) const [inline]

Is the iterator dereferenceable?

Definition at line 420 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_before_begin(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end(), and __gnu_debug::_Safe_iterator_base::_M_singular().

Referenced by __gnu_debug::__check_dereferenceable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator∗(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator->().

4.81.3.9 void __gnu_debug::_Safe_iterator_base::_M_detach ( ) [inherited]

Detach the iterator for whatever sequence it is attached to, if any.

4.81.3.10 void __gnu_debug::_Safe_iterator_base::_M_detach_single ( ) throw () [inherited]

Likewise, but not thread-safe.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.81.3.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_iterator.

4.81.3.12 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable ( ) const [inline]

Is the iterator incrementable?

Definition at line 437 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end(), and __gnu_debug::_Safe_iterator_base::_M_singular().

Referenced by __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_before_dereferenceable(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++().

4.81.3.13 void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_version.

4.81.3.14 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_before_begin ( ) const [inline]

Is this iterator equal to the sequence's before_begin() iterator if any?

Definition at line 468 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base().

Referenced by __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable().

4.81.3.15 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin ( ) const [inline]

Is this iterator equal to the sequence's begin() iterator?

Definition at line 459 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base().

4.81.3.16 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_beginnest ( ) const [inline]

Is this iterator equal to the sequence's before_begin() iterator if any or begin() otherwise?

Definition at line 475 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base().

4.81.3.17 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end ( ) const [inline]

Is this iterator equal to the sequence's end() iterator?

Definition at line 463 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base().

Referenced by __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable().

4.81.3.18 void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw () [inherited]

Reset all member variables

4.81.3.19 bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw () [inherited]

Is this iterator singular?

Referenced by __gnu_debug::__check_singular(), __gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.81.3.20 void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw () [inline], [inherited]

Unlink itself

Definition at line 151 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_next, and __gnu_debug::_Safe_iterator_base::_M_prior.

4.81.3.21 template<typename _Iterator, typename _Sequence> _Iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::base ( ) const [inline]

Return the underlying iterator.

Definition at line 396 of file safe_iterator.h.

Referenced by __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_before_dereferenceable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_before_begin(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_begin(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_beginnest(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_is_end().

4.81.3.22 template<typename _Iterator, typename _Sequence> __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator _Iterator ( ) const [inline]

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 402 of file safe_iterator.h.

4.81.3.23 template<typename _Iterator, typename _Sequence> reference __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator∗ ( ) const [inline]

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 260 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable().

4.81.3.24 template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++ ( ) [inline]

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 289 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable().

4.81.3.25 template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator++ (int) [inline]

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 303 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable().

4.81.3.26 template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- ( ) [inline]

Iterator predecrement.

Precondition

iterator is decrementable

Definition at line 319 of file safe_iterator.h.

4.81.3.27 template<typename _Iterator, typename _Sequence> _Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-- (int) [inline]

Iterator postdecrement.

Precondition

iterator is decrementable

Definition at line 333 of file safe_iterator.h.

4.81.3.28 template<typename _Iterator, typename _Sequence> pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator-> ( ) const [inline]

Iterator dereference.

Precondition

iterator is dereferenceable

Todo Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

Definition at line 275 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable().

4.81.3.29 template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator= (const _Safe_iterator< _Iterator, _Sequence > & __x) [inline]

Copy assignment.

Definition at line 217 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach(), __gnu_debug::_Safe_iterator_base::_M_sequence, and __gnu_debug::_Safe_iterator_base::_M_singular().

4.81.3.30 template<typename _Iterator, typename _Sequence> _Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator= (_Safe_iterator< _Iterator, _Sequence > && __x) [inline]

Move assignment.

Postcondition

__x is singular and unattached

Definition at line 237 of file safe_iterator.h.

References __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach().

4.81.4 Member Data Documentation

4.81.4.1 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 72 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and __gnu_debug::_Safe_iterator_base::_M_unlink().

4.81.4.2 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 68 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and __gnu_debug::_Safe_iterator_base::_M_unlink().

4.81.4.3 _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 55 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator_base::_Safe_iterator_base(), __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.81.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version `[inherited]`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Definition at line 64 of file safe_base.h.

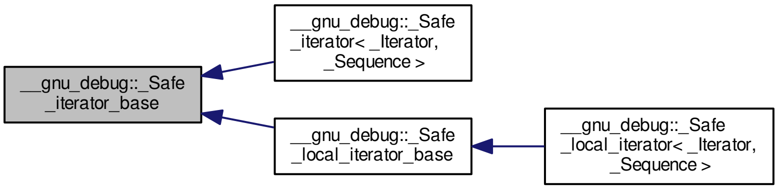
Referenced by __gnu_debug::_Safe_iterator_base::_M_invalidate().

The documentation for this class was generated from the following files:

- safe_iterator.h
- safe_iterator.tcc

4.82 __gnu_debug::_Safe_iterator_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- void _M_attach (_Safe_sequence_base ∗__seq, bool __constant)
- void _M_attach_single (_Safe_sequence_base ∗__seq, bool __constant) throw ()
- bool _M_attached_to (const _Safe_sequence_base ∗__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const throw ()
- void _M_detach ()
- void _M_detach_single () throw ()
- void _M_invalidate ()
- void _M_reset () throw ()
- bool _M_singular () const throw ()
- void _M_unlink () throw ()

Public Attributes

- _Safe_iterator_base ∗ _M_next

- _Safe_iterator_base ∗ _M_prior
- _Safe_sequence_base ∗ _M_sequence
- unsigned int _M_version

Protected Member Functions

- _Safe_iterator_base ()
- _Safe_iterator_base (const _Safe_sequence_base ∗__seq, bool __constant)
- _Safe_iterator_base (const _Safe_iterator_base &__x, bool __constant)
- **_Safe_iterator_base** (const _Safe_iterator_base &)
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- _Safe_iterator_base & **operator=** (const _Safe_iterator_base &)

4.82.1 Detailed Description

Basic functionality for a *safe* iterator.

The _Safe_iterator_base base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file safe_base.h.

4.82.2 Constructor & Destructor Documentation

4.82.2.1 __gnu_debug::_Safe_iterator_base::_Safe_iterator_base ( ) `[inline]`, `[protected]`

Initializes the iterator and makes it singular.

Definition at line 76 of file safe_base.h.

4.82.2.2 __gnu_debug::_Safe_iterator_base::_Safe_iterator_base (const **_Safe_sequence_base** ∗ *__seq,* bool *__constant*) `[inline]`, `[protected]`

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 87 of file safe_base.h.

References _M_attach().

4.82.2.3 __gnu_debug::_Safe_iterator_base::_Safe_iterator_base (const **_Safe_iterator_base** & *__x,* bool *__constant*) `[inline]`, `[protected]`

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 94 of file safe_base.h.

References _M_attach(), and _M_sequence.

4.82.3 Member Function Documentation

4.82.3.1 void __gnu_debug::_Safe_iterator_base::_M_attach (_Safe_sequence_base ∗ __seq, bool __constant)

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach(), and _Safe_iterator_base().

4.82.3.2 void __gnu_debug::_Safe_iterator_base::_M_attach_single (_Safe_sequence_base ∗ __seq, bool __constant) throw ()

Likewise, but not thread-safe.

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach_single(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_attach_single().

4.82.3.3 bool __gnu_debug::_Safe_iterator_base::_M_attached_to (const _Safe_sequence_base ∗ __seq) const [inline]

Determines if we are attached to the given sequence.

Definition at line 129 of file safe_base.h.

References _M_sequence.

4.82.3.4 bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw ()

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.82.3.5 void __gnu_debug::_Safe_iterator_base::_M_detach ( )

Detach the iterator for whatever sequence it is attached to, if any.

4.82.3.6 void __gnu_debug::_Safe_iterator_base::_M_detach_single ( ) throw ()

Likewise, but not thread-safe.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.82.3.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw () [protected]

For use in _Safe_iterator.

4.82.3.8 void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline]

Invalidate the iterator, making it singular.

Definition at line 142 of file safe_base.h.

References _M_version.

4.82.3.9 void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw ()

Reset all member variables

4.82.3.10 bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw ()

Is this iterator singular?

Referenced by __gnu_debug::__check_singular(), __gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_local-

_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.82.3.11 void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw () [inline]

Unlink itself

Definition at line 151 of file safe_base.h.

References _M_next, and _M_prior.

4.82.4 Member Data Documentation

4.82.4.1 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 72 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

4.82.4.2 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 68 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

4.82.4.3 _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 55 of file safe_base.h.

Referenced by _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), _Safe_iterator_base(), __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.82.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Definition at line 64 of file safe_base.h.

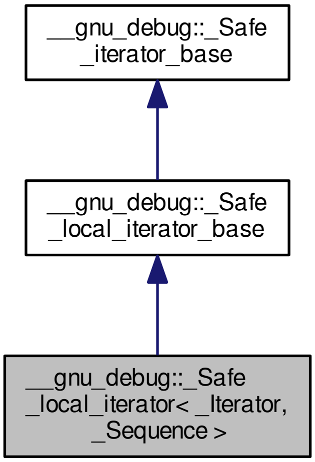
Referenced by _M_invalidate().

The documentation for this class was generated from the following file:

- safe_base.h

4.83 __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence > Class Template Reference

Inheritance diagram for __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >:



Public Types

- typedef _Traits::difference_type **difference_type**
- typedef _Traits::iterator_category **iterator_category**
- typedef _Iterator **iterator_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::value_type **value_type**

Public Member Functions

- _Safe_local_iterator ()
- _Safe_local_iterator (const _Iterator &__i, size_type __bucket, const _Sequence ∗__seq)
- _Safe_local_iterator (const _Safe_local_iterator &__x)
- template< typename _MutableIterator >
  _Safe_local_iterator (const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< std::__are_same< _MutableIterator, typename _Sequence::local_iterator::iterator_type >::__value, _Sequence >::__type > &__x)
- void _M_attach (_Safe_sequence_base ∗__seq, bool __constant)
- void _M_attach (_Safe_sequence_base ∗__seq)
- void _M_attach_single (_Safe_sequence_base ∗__seq, bool __constant) throw ()
- void _M_attach_single (_Safe_sequence_base ∗__seq)
- bool _M_attached_to (const _Safe_sequence_base ∗__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const throw ()
- bool _M_dereferenceable () const

- void _M_detach ()
- void _M_detach_single () throw ()
- const _Sequence ∗ **_M_get_sequence** () const
- template<typename _Other >
  bool _M_in_same_bucket (const _Safe_local_iterator< _Other, _Sequence > &__other) const
- bool _M_incrementable () const
- void _M_invalidate ()
- bool _M_is_begin () const
- bool _M_is_end () const
- void _M_reset () throw ()
- bool _M_singular () const throw ()
- void _M_unlink () throw ()
- template<typename _Other >
  bool **_M_valid_range** (const _Safe_local_iterator< _Other, _Sequence > &__rhs) const
- _Iterator base () const
- size_type bucket () const
- operator _Iterator () const
- reference operator∗ () const
- _Safe_local_iterator & operator++ ()
- _Safe_local_iterator operator++ (int)
- pointer operator-> () const
- _Safe_local_iterator & operator= (const _Safe_local_iterator &__x)

Public Attributes

- _Safe_iterator_base ∗ _M_next
- _Safe_iterator_base ∗ _M_prior
- _Safe_sequence_base ∗ _M_sequence
- unsigned int _M_version

Protected Member Functions

- _Safe_unordered_container_base ∗ **_M_get_container** () const noexcept
- __gnu_cxx::__mutex & _M_get_mutex () throw ()

4.83.1 Detailed Description

template<typename _Iterator, typename _Sequence>class __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >

Safe iterator wrapper.

The class template _Safe_local_iterator is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In additional to the basic iterator operations (which are validated, and then passed to the underlying iterator), _Safe_local_iterator has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 52 of file safe_local_iterator.h.

4.83.2 Constructor & Destructor Documentation

4.83.2.1 template<typename _Iterator, typename _Sequence> **__gnu_debug::_Safe_local_iterator**< _Iterator, _Sequence >::**_Safe_local_iterator** ( ) `[inline]`

Postcondition

the iterator is singular and unattached

Definition at line 82 of file safe_local_iterator.h.

4.83.2.2 template<typename _Iterator, typename _Sequence> **__gnu_debug::_Safe_local_iterator**< _Iterator, _Sequence >::**_Safe_local_iterator** (const _Iterator & *__i,* size_type *__bucket,* const _Sequence * *__seq*) `[inline]`

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

Definition at line 91 of file safe_local_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_singular().

4.83.2.3 template<typename _Iterator, typename _Sequence> **__gnu_debug::_Safe_local_iterator**< _Iterator, _Sequence >::**_Safe_local_iterator** (const **_Safe_local_iterator**< _Iterator, _Sequence > & *__x*) `[inline]`

Copy construction.

Definition at line 104 of file safe_local_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_singular().

4.83.2.4 template<typename _Iterator, typename _Sequence> template<typename _MutableIterator > **__gnu_debug::_Safe_local_iterator**< _Iterator, _Sequence >::**_Safe_local_iterator** (const **_Safe_local_iterator**< _MutableIterator, typename __gnu_cxx::__enable_if< std::__are_same< _MutableIterator, typename _Sequence::local_iterator::iterator_type >::__value, _Sequence >::__type > & *__x*) `[inline]`

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 122 of file safe_local_iterator.h.

4.83.3 Member Function Documentation

4.83.3.1 void __gnu_debug::_Safe_local_iterator_base::_M_attach (**_Safe_sequence_base** * *__seq,* bool *__constant*) `[inherited]`

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base().

4.83.3.2 template<typename _Iterator, typename _Sequence> void __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach (_Safe_sequence_base * __seq) [inline]

Attach iterator to the given sequence.

Definition at line 238 of file safe_local_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_attach().

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=().

4.83.3.3 void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw () [inherited]

Likewise, but not thread-safe.

4.83.3.4 template<typename _Iterator, typename _Sequence> void __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach_single (_Safe_sequence_base * __seq) [inline]

Likewise, but not thread-safe.

Definition at line 243 of file safe_local_iterator.h.

References __gnu_debug::_Safe_iterator_base::_M_attach_single().

4.83.3.5 bool __gnu_debug::_Safe_iterator_base::_M_attached_to (const _Safe_sequence_base * __seq) const [inline], [inherited]

Determines if we are attached to the given sequence.

Definition at line 129 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_sequence.

4.83.3.6 bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw () [inherited]

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.83.3.7 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable ( ) const [inline]

Is the iterator dereferenceable?

Definition at line 248 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end(), and __gnu_debug::_Safe_iterator_base::_M_singular().

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator*(), and __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator->().

4.83.3.8 void __gnu_debug::_Safe_local_iterator_base::_M_detach ( ) [inherited]

Detach the iterator for whatever container it is attached to, if any.

4.83.3.9 void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ( ) throw () [inherited]

Likewise, but not thread-safe.

4.83.3.10 __gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_iterator.

4.83.3.11 template<typename _Iterator, typename _Sequence> template<typename _Other > bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_in_same_bucket (const _Safe_local_iterator< _Other, _Sequence > & *__other*) const [inline]

Is this iterator part of the same bucket as the other one?

Definition at line 277 of file safe_local_iterator.h.

4.83.3.12 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable ( ) const [inline]

Is the iterator incrementable?

Definition at line 253 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end(), and __gnu_debug::_Safe_iterator_base::_M_singular().

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++().

4.83.3.13 void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_version.

4.83.3.14 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_begin ( ) const [inline]

Is this iterator equal to the sequence's begin() iterator?

Definition at line 268 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base().

4.83.3.15 template<typename _Iterator, typename _Sequence> bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end ( ) const [inline]

Is this iterator equal to the sequence's end() iterator?

Definition at line 272 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base().

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable(), and __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable().

4.83.3.16 void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw () [inherited]

Reset all member variables

4.83.3.17 bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw () [inherited]

Is this iterator singular?

Referenced by __gnu_debug::__check_singular(), __gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.83.3.18 void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw () [inline], [inherited]

Unlink itself

Definition at line 151 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_next, and __gnu_debug::_Safe_iterator_base::_M_prior.

4.83.3.19 template< typename _Iterator, typename _Sequence > _Iterator __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base ( ) const [inline]

Return the underlying iterator.

Definition at line 222 of file safe_local_iterator.h.

Referenced by __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_begin(), and __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end().

4.83.3.20 template< typename _Iterator, typename _Sequence > size_type __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket ( ) const [inline]

Return the bucket.

Definition at line 228 of file safe_local_iterator.h.

4.83.3.21 template< typename _Iterator, typename _Sequence > __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator _Iterator ( ) const [inline]

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 234 of file safe_local_iterator.h.

4.83.3.22 template< typename _Iterator, typename _Sequence > reference __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator* ( ) const [inline]

Iterator dereference.

Precondition

iterator is dereferenceable

Definition at line 164 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable().

4.83.3.23 template< typename _Iterator, typename _Sequence > _Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ ( ) [inline]

Iterator preincrement.

Precondition

iterator is incrementable

Definition at line 193 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable().

4.83.3.24 template<typename _Iterator, typename _Sequence> _Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ (int) [inline]

Iterator postincrement.

Precondition

iterator is incrementable

Definition at line 207 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable().

4.83.3.25 template<typename _Iterator, typename _Sequence> pointer __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator-> ( ) const [inline]

Iterator dereference.

Precondition

iterator is dereferenceable

Todo Make this correct w.r.t. iterators that return proxies

Use addressof() instead of & operator

Definition at line 179 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable().

4.83.3.26 template<typename _Iterator, typename _Sequence> _Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator= (const _Safe_local_iterator< _Iterator, _Sequence > & __x) [inline]

Copy assignment.

Definition at line 144 of file safe_local_iterator.h.

References __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach(), __gnu_debug::_Safe_iterator_base::_M_sequence, and __gnu_debug::_Safe_iterator_base::_M_singular().

4.83.4 Member Data Documentation

4.83.4.1 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 72 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and __gnu_debug::_Safe_iterator_base::_M_unlink().

4.83.4.2 _Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior `[inherited]`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 68 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and __gnu_debug::_Safe_iterator_base::_M_unlink().

4.83.4.3 _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence `[inherited]`

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 55 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator_base::_Safe_iterator_base(), __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.83.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version `[inherited]`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Definition at line 64 of file safe_base.h.

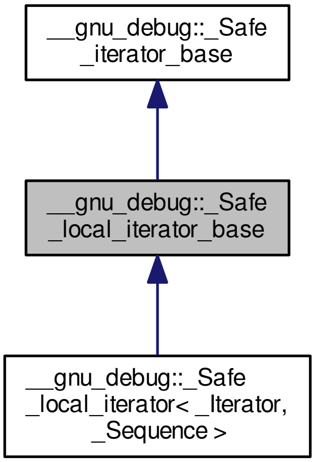
Referenced by __gnu_debug::_Safe_iterator_base::_M_invalidate().

The documentation for this class was generated from the following files:

- safe_local_iterator.h
- safe_local_iterator.tcc

4.84 __gnu_debug::_Safe_local_iterator_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_local_iterator_base:



Public Member Functions

- void _M_attach (_Safe_sequence_base ∗__seq, bool __constant)
- void _M_attach_single (_Safe_sequence_base ∗__seq, bool __constant) throw ()
- bool _M_attached_to (const _Safe_sequence_base ∗__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const throw ()
- void _M_detach ()
- void _M_detach_single () throw ()
- void _M_invalidate ()
- void _M_reset () throw ()
- bool _M_singular () const throw ()
- void _M_unlink () throw ()

Public Attributes

- _Safe_iterator_base ∗ _M_next
- _Safe_iterator_base ∗ _M_prior
- _Safe_sequence_base ∗ _M_sequence
- unsigned int _M_version

Protected Member Functions

- _Safe_local_iterator_base ()
- _Safe_local_iterator_base (const _Safe_sequence_base ∗__seq, bool __constant)

- _Safe_local_iterator_base (const _Safe_local_iterator_base &__x, bool __constant)
- **_Safe_local_iterator_base** (const _Safe_local_iterator_base &)
- _Safe_unordered_container_base ∗ **_M_get_container** () const noexcept
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- _Safe_local_iterator_base & **operator=** (const _Safe_local_iterator_base &)

4.84.1 Detailed Description

Basic functionality for a *safe* iterator.

The _Safe_local_iterator_base base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file safe_unordered_base.h.

4.84.2 Constructor & Destructor Documentation

4.84.2.1 __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ( ) `[inline]`, `[protected]`

Initializes the iterator and makes it singular.

Definition at line 54 of file safe_unordered_base.h.

4.84.2.2 __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (const **_Safe_sequence_base** ∗ *__seq*, bool *__constant*) `[inline]`, `[protected]`

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 64 of file safe_unordered_base.h.

References _M_attach().

4.84.2.3 __gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (const **_Safe_local_iterator_base** & *__x*, bool *__constant*) `[inline]`, `[protected]`

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 70 of file safe_unordered_base.h.

References _M_attach(), and __gnu_debug::_Safe_iterator_base::_M_sequence.

4.84.3 Member Function Documentation

4.84.3.1 void __gnu_debug::_Safe_local_iterator_base::_M_attach (**_Safe_sequence_base** ∗ *__seq*, bool *__constant*)

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by _Safe_local_iterator_base().

4.84.3.2 void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (_Safe_sequence_base ∗ __seq, bool __constant) throw ()

Likewise, but not thread-safe.

4.84.3.3 bool __gnu_debug::_Safe_iterator_base::_M_attached_to (const _Safe_sequence_base ∗ __seq) const [inline], [inherited]

Determines if we are attached to the given sequence.

Definition at line 129 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_sequence.

4.84.3.4 bool __gnu_debug::_Safe_iterator_base::_M_can_compare (const _Safe_iterator_base & __x) const throw () [inherited]

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

4.84.3.5 void __gnu_debug::_Safe_local_iterator_base::_M_detach ( )

Detach the iterator for whatever container it is attached to, if any.

4.84.3.6 void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ( ) throw ()

Likewise, but not thread-safe.

4.84.3.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_iterator.

4.84.3.8 void __gnu_debug::_Safe_iterator_base::_M_invalidate ( ) [inline], [inherited]

Invalidate the iterator, making it singular.

Definition at line 142 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_version.

4.84.3.9 void __gnu_debug::_Safe_iterator_base::_M_reset ( ) throw () [inherited]

Reset all member variables

4.84.3.10 bool __gnu_debug::_Safe_iterator_base::_M_singular ( ) const throw () [inherited]

Is this iterator singular?

Referenced by __gnu_debug::__check_singular(), __gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_dereferenceable(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.84.3.11 void __gnu_debug::_Safe_iterator_base::_M_unlink ( ) throw () [inline], [inherited]

Unlink itself

Definition at line 151 of file safe_base.h.

References __gnu_debug::_Safe_iterator_base::_M_next, and __gnu_debug::_Safe_iterator_base::_M_prior.

4.84.4 Member Data Documentation

4.84.4.1 _Safe_iterator_base∗ __gnu_debug::_Safe_iterator_base::_M_next `[inherited]`

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 72 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and __gnu_debug::_Safe_iterator_base::_M_unlink().

4.84.4.2 _Safe_iterator_base∗ __gnu_debug::_Safe_iterator_base::_M_prior `[inherited]`

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Definition at line 68 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and __gnu_debug::_Safe_iterator_base::_M_unlink().

4.84.4.3 _Safe_sequence_base∗ __gnu_debug::_Safe_iterator_base::_M_sequence `[inherited]`

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 55 of file safe_base.h.

Referenced by __gnu_debug::_Safe_iterator_base::_M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator_base::_Safe_iterator_base(), _Safe_local_iterator_base(), __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator=(), and __gnu_debug::_Safe_iterator< _Iterator, _Sequence >::operator=().

4.84.4.4 unsigned int __gnu_debug::_Safe_iterator_base::_M_version `[inherited]`

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Definition at line 64 of file safe_base.h.

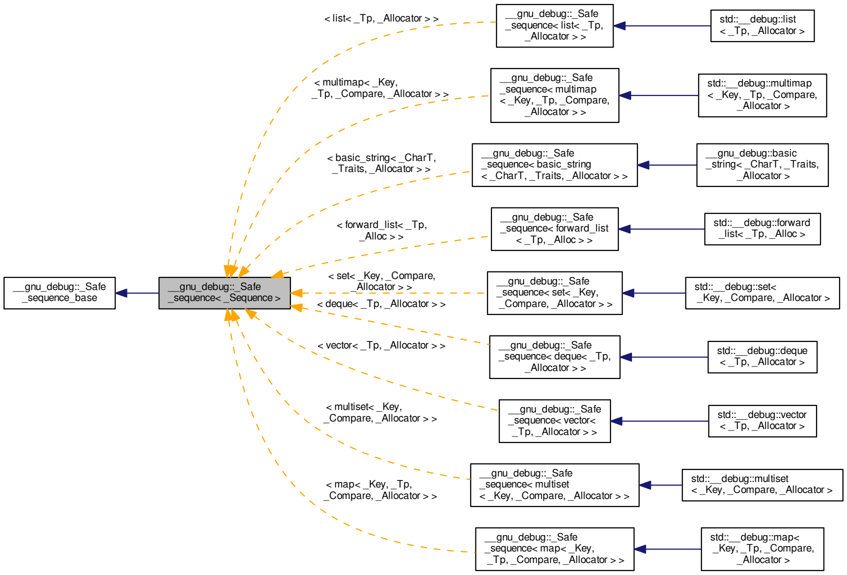
Referenced by __gnu_debug::_Safe_iterator_base::_M_invalidate().

The documentation for this class was generated from the following file:

- safe_unordered_base.h

4.85 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference

Inheritance diagram for __gnu_debug::_Safe_sequence< _Sequence >:



Public Member Functions

- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- template<typename _Predicate >
  void _M_invalidate_if (_Predicate __pred)
- template<typename _Predicate >
  void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.85.1 Detailed Description

template<typename _Sequence>class __gnu_debug::_Safe_sequence< _Sequence >

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template _Safe_sequence simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template _Safe_iterator for this sequence. Iterators will then be tracked automatically.

Definition at line 111 of file safe_sequence.h.

4.85.2 Member Function Documentation

4.85.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) `[inherited]`

Attach an iterator to this sequence.

4.85.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw () `[inherited]`

Likewise but not thread safe.

4.85.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it) `[inherited]`

Detach an iterator from this sequence

4.85.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) `[protected]`, `[inherited]`

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.85.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw () `[inherited]`

Likewise but not thread safe.

4.85.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) `[protected]`, `[inherited]`

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.85.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () `[protected]`, `[inherited]`

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.85.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.85.2.9 template<typename _Sequence > template<typename _Predicate > void _Safe_sequence< _Sequence >::_M_invalidate_if (_Predicate __pred)

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe_sequence.tcc.

4.85.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.85.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.85.2.12 template<typename _Sequence > template<typename _Predicate > void _Safe_sequence< _Sequence >::_M_transfer_from_if (_Safe_sequence< _Sequence > & __from, _Predicate __pred)

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe_sequence.tcc.

References __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_iterator_base::_M_detach_single(), __gnu_debug::_Safe_sequence_base::_M_get_mutex(), __gnu_debug::_Safe_sequence_base::_M_iterators, __gnu_debug::_Safe_iterator_base::_M_next, __gnu_debug::_Safe_iterator_base::_M_prior, __gnu_debug::_Safe_iterator_base::_M_sequence, and __gnu_debug::_Safe_sequence_base::_M_version.

4.85.3 Member Data Documentation

4.85.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.85.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.85.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

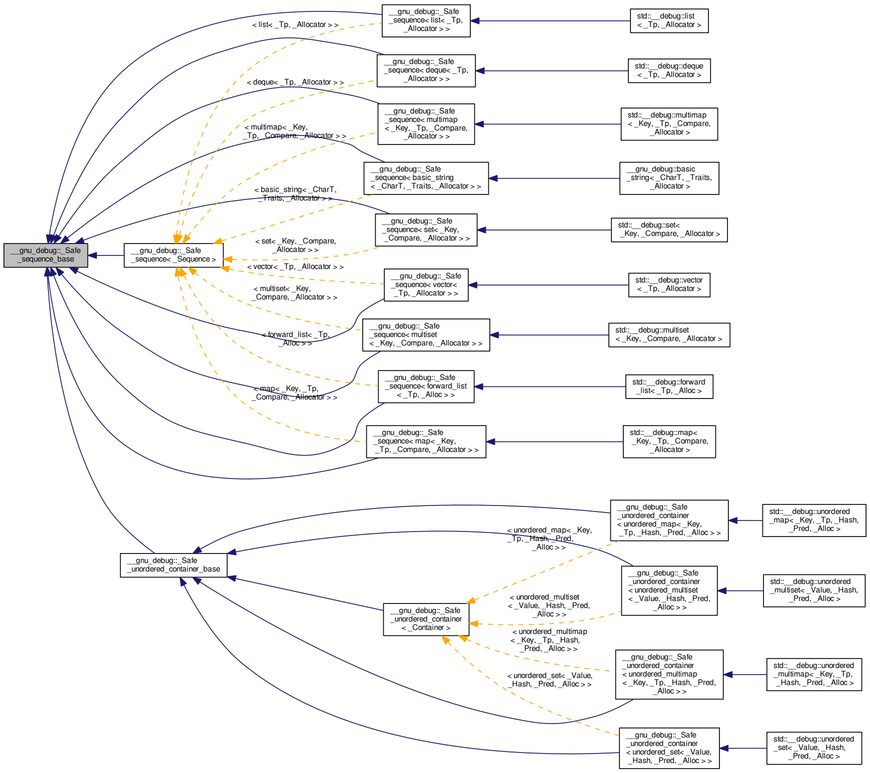
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following files:

- safe_sequence.h
- safe_sequence.tcc

4.86 __gnu_debug::_Safe_sequence_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_sequence_base:



Public Member Functions

- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- ∼_Safe_sequence_base ()
- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.86.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The _Safe_sequence_base class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from _Safe_sequence_base publicly, so that safe iterators (which inherit _Safe_iterator_base) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 177 of file safe_base.h.

4.86.2 Constructor & Destructor Documentation

4.86.2.1 __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base ( ) [inline], [protected]

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 197 of file safe_base.h.

References _M_detach_all().

4.86.3 Member Function Documentation

4.86.3.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ *__it,* bool *__constant*)

Attach an iterator to this sequence.

4.86.3.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw ()

Likewise but not thread safe.

4.86.3.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ *__it*)

Detach an iterator from this sequence

4.86.3.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected]

Detach all iterators, leaving them singular.

Referenced by ∼_Safe_sequence_base().

4.86.3.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ *__it*) throw ()

Likewise but not thread safe.

4.86.3.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.86.3.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.86.3.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References _M_version.

4.86.3.9 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.86.3.10 void __gnu_debug::_Safe_sequence_base::_M_swap (**_Safe_sequence_base** & *__x*) [protected]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.86.4 Member Data Documentation

4.86.4.1 **_Safe_iterator_base**∗ __gnu_debug::_Safe_sequence_base::_M_const_iterators

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.86.4.2 **_Safe_iterator_base**∗ __gnu_debug::_Safe_sequence_base::_M_iterators

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.86.4.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

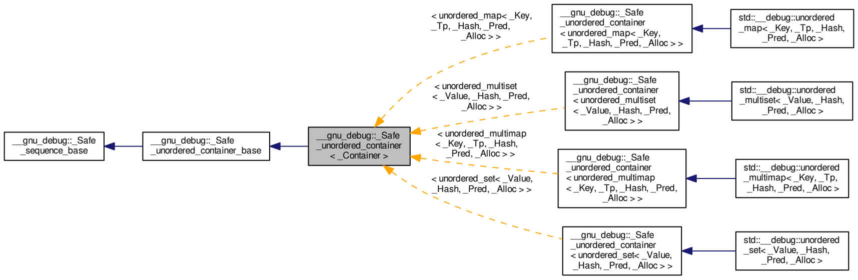
Referenced by _M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- safe_base.h

4.87 __gnu_debug::_Safe_unordered_container< _Container > Class Template Reference

Inheritance diagram for __gnu_debug::_Safe_unordered_container< _Container >:



Public Member Functions

- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_local (_Safe_iterator_base ∗__it)
- void _M_detach_local_single (_Safe_iterator_base ∗__it) throw ()
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- template<typename _Predicate >
  void _M_invalidate_if (_Predicate __pred)
- template<typename _Predicate >
  void _M_invalidate_local_if (_Predicate __pred)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_const_local_iterators
- _Safe_iterator_base ∗ _M_iterators
- _Safe_iterator_base ∗ _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_unordered_container_base &__x)
- void _M_swap (_Safe_sequence_base &__x)

4.87.1 Detailed Description

template<typename _Container>class __gnu_debug::_Safe_unordered_container< _Container >

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template _Safe_unordered_container simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template _Safe_iterator for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template _Safe_local_iterator for this container. Iterators will then be tracked automatically.

Definition at line 58 of file safe_unordered_container.h.

4.87.2 Member Function Documentation

4.87.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (**_Safe_iterator_base** * *__it,* bool *__constant*) `[inherited]`

Attach an iterator to this sequence.

4.87.2.2 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (**_Safe_iterator_base** * *__it,* bool *__constant*) `[inherited]`

Attach an iterator to this container.

4.87.2.3 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (**_Safe_iterator_base** * *__it,* bool *__constant*) throw () `[inherited]`

Likewise but not thread safe.

4.87.2.4 void __gnu_debug::_Safe_sequence_base::_M_attach_single (**_Safe_iterator_base** * *__it,* bool *__constant*) throw () `[inherited]`

Likewise but not thread safe.

4.87.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach (**_Safe_iterator_base** * *__it*) `[inherited]`

Detach an iterator from this sequence

4.87.2.6 void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) `[protected]`, `[inherited]`

Detach all iterators, leaving them singular.

4.87.2.7 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (**_Safe_iterator_base** * *__it*) `[inherited]`

Detach an iterator from this container

4.87.2.8 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (**_Safe_iterator_base** * *__it*) throw () `[inherited]`

Likewise but not thread safe.

4.87.2.9 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.87.2.10 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.87.2.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.87.2.12 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.87.2.13 template<typename _Container > template<typename _Predicate > void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_if (_Predicate __pred)

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe_unordered_container.tcc.

4.87.2.14 template<typename _Container > template<typename _Predicate > void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if (_Predicate __pred)

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

Definition at line 70 of file safe_unordered_container.tcc.

4.87.2.15 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.87.2.16 void __gnu_debug::_Safe_unordered_container_base::_M_swap (_Safe_unordered_container_base & __x) [protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.87.2.17 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.87.3 Member Data Documentation

4.87.3.1 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.87.3.2 _Safe_iterator_base∗ __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.87.3.3 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.87.3.4 _Safe_iterator_base∗ __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.87.3.5 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

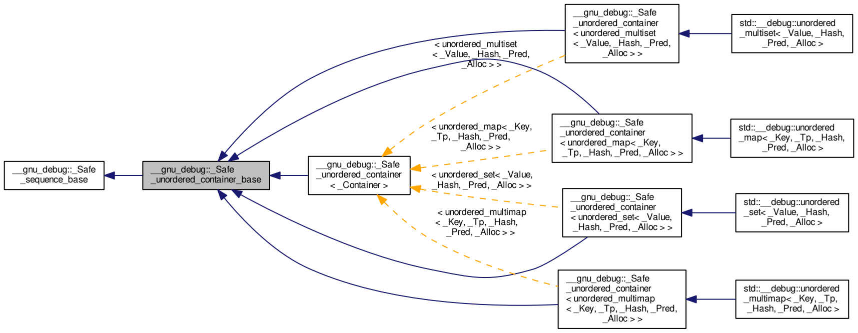
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following files:

- safe_unordered_container.h
- safe_unordered_container.tcc

4.88 __gnu_debug::_Safe_unordered_container_base Class Reference

Inheritance diagram for __gnu_debug::_Safe_unordered_container_base:



Public Member Functions

- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_local (_Safe_iterator_base ∗__it)
- void _M_detach_local_single (_Safe_iterator_base ∗__it) throw ()
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_const_local_iterators
- _Safe_iterator_base ∗ _M_iterators
- _Safe_iterator_base ∗ _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- ∼_Safe_unordered_container_base ()
- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_unordered_container_base &__x)
- void _M_swap (_Safe_sequence_base &__x)

4.88.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The _Safe_unordered_container_base class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from _Safe_unordered_container_base publicly, so that safe iterators (which inherit _Safe_iterator_base) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 123 of file safe_unordered_base.h.

4.88.2 Constructor & Destructor Documentation

4.88.2.1 __gnu_debug::_Safe_unordered_container_base::∼_Safe_unordered_container_base ( ) `[inline]`, `[protected]`

Notify all iterators that reference this container that the container is being destroyed.

Definition at line 141 of file safe_unordered_base.h.

4.88.3 Member Function Documentation

4.88.3.1 void __gnu_debug::_Safe_sequence_base::_M_attach (**_Safe_iterator_base** ∗ *__it,* bool *__constant*) `[inherited]`

Attach an iterator to this sequence.

4.88.3.2 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (**_Safe_iterator_base** ∗ *__it,* bool *__constant*)

Attach an iterator to this container.

4.88.3.3 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (**_Safe_iterator_base** ∗ *__it,* bool *__constant*) throw ()

Likewise but not thread safe.

4.88.3.4 void __gnu_debug::_Safe_sequence_base::_M_attach_single (**_Safe_iterator_base** ∗ *__it,* bool *__constant*) throw () `[inherited]`

Likewise but not thread safe.

4.88.3.5 void __gnu_debug::_Safe_sequence_base::_M_detach (**_Safe_iterator_base** ∗ *__it*) `[inherited]`

Detach an iterator from this sequence

4.88.3.6 void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) `[protected]`

Detach all iterators, leaving them singular.

4.88.3.7 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (**_Safe_iterator_base** ∗ *__it*)

Detach an iterator from this container

4.88.3.8 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (**_Safe_iterator_base** ∗ *__it*) throw ()

Likewise but not thread safe.

4.88.3.9 void __gnu_debug::_Safe_sequence_base::_M_detach_single (**_Safe_iterator_base** ∗ *__it*) throw () `[inherited]`

Likewise but not thread safe.

4.88.3.10 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) `[protected]`, `[inherited]`

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.88.3.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () `[protected]`, `[inherited]`

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.88.3.12 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.88.3.13 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.88.3.14 void __gnu_debug::_Safe_unordered_container_base::_M_swap (**_Safe_unordered_container_base** & *__x*) `[protected]`

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.88.3.15 void __gnu_debug::_Safe_sequence_base::_M_swap (**_Safe_sequence_base** & *__x*) `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.88.4 Member Data Documentation

4.88.4.1 **_Safe_iterator_base**∗ __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.88.4.2 _Safe_iterator_base∗ __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.88.4.3 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.88.4.4 _Safe_iterator_base∗ __gnu_debug::_Safe_unordered_container_base::_M_local_iterators

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.88.4.5 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

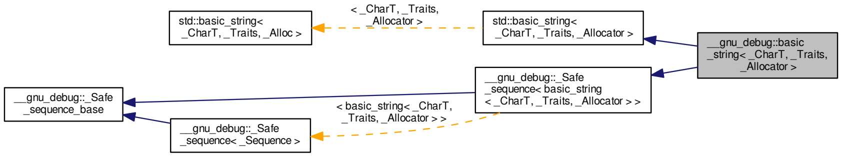
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- safe_unordered_base.h

4.89 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference

Inheritance diagram for __gnu_debug::basic_string< _CharT, _Traits, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator < typename _Base::const_iterator, basic_string > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**

- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator < typename _Base::iterator, basic_string > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **basic_string** (const _Allocator &__a=_Allocator())
- **basic_string** (const _Base &__base)
- **basic_string** (const basic_string &__str)
- **basic_string** (const basic_string &__str, size_type __pos, size_type __n=_Base::npos, const _Allocator &__a=-_Allocator())
- **basic_string** (const _CharT ∗__s, size_type __n, const _Allocator &__a=_Allocator())
- **basic_string** (const _CharT ∗__s, const _Allocator &__a=_Allocator())
- **basic_string** (size_type __n, _CharT __c, const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
  basic_string (_InputIterator __begin, _InputIterator __end, const _Allocator &__a=_Allocator())
- **basic_string** (basic_string &&__str) noexcept
- **basic_string** (std::initializer_list< _CharT > __l, const _Allocator &__a=_Allocator())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- basic_string & **append** (const basic_string &__str)
- basic_string & **append** (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & **append** (const _CharT ∗__s, size_type __n)
- basic_string & **append** (const _CharT ∗__s)
- basic_string & **append** (size_type __n, _CharT __c)
- template<typename _InputIterator >
  basic_string & **append** (_InputIterator __first, _InputIterator __last)
- basic_string & append (const basic_string &__str)
- basic_string & append (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & append (initializer_list< _CharT > __l)
- basic_string & **assign** (const basic_string &__x)
- basic_string & **assign** (basic_string &&__x)
- basic_string & **assign** (const basic_string &__str, size_type __pos, size_type __n)

- basic_string & **assign** (const _CharT ∗__s, size_type __n)
- basic_string & **assign** (const _CharT ∗__s)
- basic_string & **assign** (size_type __n, _CharT __c)
- template<typename _InputIterator >
  basic_string & **assign** (_InputIterator __first, _InputIterator __last)
- basic_string & **assign** (std::initializer_list< _CharT > __l)
- basic_string & assign (const basic_string &__str)
- basic_string & assign (basic_string &&__str)
- basic_string & assign (const basic_string &__str, size_type __pos, size_type __n)
- const_reference at (size_type __n) const
- reference at (size_type __n)
- reference back ()
- const_reference back () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const _CharT ∗ **c_str** () const noexcept
- size_type capacity () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- int **compare** (const basic_string &__str) const
- int **compare** (size_type __pos1, size_type __n1, const basic_string &__str) const
- int **compare** (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2) const
- int **compare** (const _CharT ∗__s) const
- int **compare** (size_type __pos1, size_type __n1, const _CharT ∗__s) const
- int **compare** (size_type __pos1, size_type __n1, const _CharT ∗__s, size_type __n2) const
- int compare (const basic_string &__str) const
- int compare (size_type __pos, size_type __n, const basic_string &__str) const
- int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2) const
- size_type **copy** (_CharT ∗__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- const _CharT ∗ **data** () const noexcept
- bool empty () const noexcept
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- basic_string & **erase** (size_type __pos=0, size_type __n=_Base::npos)
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- size_type **find** (const basic_string &__str, size_type __pos=0) const noexcept
- size_type **find** (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type **find** (const _CharT ∗__s, size_type __pos=0) const
- size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- size_type **find_first_not_of** (const basic_string &__str, size_type __pos=0) const noexcept
- size_type **find_first_not_of** (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type **find_first_not_of** (const _CharT ∗__s, size_type __pos=0) const

- size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type **find_first_of** (const basic_string &__str, size_type __pos=0) const noexcept
- size_type **find_first_of** (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type **find_first_of** (const _CharT ∗__s, size_type __pos=0) const
- size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type **find_last_not_of** (const basic_string &__str, size_type __pos=_Base::npos) const noexcept
- size_type **find_last_not_of** (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type **find_last_not_of** (const _CharT ∗__s, size_type __pos=_Base::npos) const
- size_type **find_last_not_of** (_CharT __c, size_type __pos=_Base::npos) const noexcept
- size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type **find_last_of** (const basic_string &__str, size_type __pos=_Base::npos) const noexcept
- size_type **find_last_of** (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type **find_last_of** (const _CharT ∗__s, size_type __pos=_Base::npos) const
- size_type **find_last_of** (_CharT __c, size_type __pos=_Base::npos) const noexcept
- size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- basic_string & **insert** (size_type __pos1, const basic_string &__str)
- basic_string & **insert** (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n)
- basic_string & **insert** (size_type __pos, const _CharT ∗__s, size_type __n)
- basic_string & **insert** (size_type __pos, const _CharT ∗__s)
- basic_string & **insert** (size_type __pos, size_type __n, _CharT __c)
- iterator **insert** (iterator __p, _CharT __c)
- void **insert** (iterator __p, size_type __n, _CharT __c)
- template<typename _InputIterator >
  void **insert** (iterator __p, _InputIterator __first, _InputIterator __last)
- void **insert** (iterator __p, std::initializer_list< _CharT > __l)
- void insert (iterator __p, size_type __n, _CharT __c)
- void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, initializer_list< _CharT > __l)
- basic_string & insert (size_type __pos1, const basic_string &__str)
- basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n)
- iterator insert (iterator __p, _CharT __c)
- size_type length () const noexcept
- size_type max_size () const noexcept
- basic_string & **operator+=** (const basic_string &__str)
- basic_string & **operator+=** (const _CharT ∗__s)
- basic_string & **operator+=** (_CharT __c)
- basic_string & **operator+=** (std::initializer_list< _CharT > __l)
- basic_string & operator+= (const basic_string &__str)
- basic_string & **operator=** (const basic_string &__str)
- basic_string & **operator=** (const _CharT ∗__s)
- basic_string & **operator=** (_CharT __c)
- basic_string & **operator=** (basic_string &&__str)
- basic_string & **operator=** (std::initializer_list< _CharT > __l)
- const_reference **operator[]** (size_type __pos) const
- reference **operator[]** (size_type __pos)
- void **pop_back** ()

- void **push_back** (_CharT __c)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- basic_string & **replace** (size_type __pos1, size_type __n1, const basic_string &__str)
- basic_string & **replace** (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2)
- basic_string & **replace** (size_type __pos, size_type __n1, const _CharT ∗__s, size_type __n2)
- basic_string & **replace** (size_type __pos, size_type __n1, const _CharT ∗__s)
- basic_string & **replace** (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- basic_string & **replace** (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT ∗__s, size_type __n)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT ∗__s)
- basic_string & **replace** (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- template< typename _InputIterator >
  basic_string & **replace** (iterator __i1, iterator __i2, _InputIterator __j1, _InputIterator __j2)
- basic_string & **replace** (iterator __i1, iterator __i2, std::initializer_list< _CharT > __l)
- basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2)
- basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT ∗__s, size_type __n)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT ∗__s)
- basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, _CharT ∗__k1, _CharT ∗__k2)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT ∗__k1, const _CharT ∗__k2)
- basic_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- void reserve (size_type __res_arg=0)
- void **resize** (size_type __n, _CharT __c)
- void **resize** (size_type __n)
- size_type **rfind** (const basic_string &__str, size_type __pos=_Base::npos) const noexcept
- size_type **rfind** (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type **rfind** (const _CharT ∗__s, size_type __pos=_Base::npos) const
- size_type **rfind** (_CharT __c, size_type __pos=_Base::npos) const noexcept
- size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- void **shrink_to_fit** ()
- size_type size () const noexcept
- basic_string **substr** (size_type __pos=0, size_type __n=_Base::npos) const
- void **swap** (basic_string< _CharT, _Traits, _Allocator > &__x)
- void swap (basic_string &__s)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Static Public Attributes

- static const size_type npos

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.89.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>class __gnu_debug::basic_string< _CharT, _Traits, _Allocator >

Class std::basic_string with safety/checking/debug instrumentation.

Definition at line 41 of file debug/string.

4.89.2 Member Function Documentation

4.89.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]

Attach an iterator to this sequence.

4.89.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.89.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.89.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.89.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw () [inherited]

Likewise but not thread safe.

4.89.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.89.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () `[protected]`, `[inherited]`

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.89.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.89.2.9 void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_invalidate_if (_Predicate __pred) `[inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.89.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.89.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.89.2.12 void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_transfer_from_if (_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > > & __from, _Predicate __pred) `[inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.89.2.13 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::append (const basic_string< _CharT, _Traits, _Allocator > & __str) `[inherited]`

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

4.89.2.14 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::append (const basic_string< _CharT, _Traits, _Allocator > & __str, size_type __pos, size_type __n) `[inherited]`

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

4.89.2.15 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::append (initializer_list< _CharT > __l) [inline], [inherited]

Append an initializer_list of characters.

Parameters

__l	The initializer_list of characters to append.

Returns

Reference to this string.

Definition at line 1030 of file basic_string.h.

4.89.2.16 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::assign (const basic_string< _CharT, _Traits, _Allocator > & __str) [inherited]

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

4.89.2.17 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::assign (basic_string< _CharT, _Traits, _Allocator > && __str) [inline], [inherited]

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

Definition at line 1079 of file basic_string.h.

4.89.2.18 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::assign (const basic_string< _CharT, _Traits, _Allocator > & *__str,* size_type *__pos,* size_type *__n*) [inline], [inherited]

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

Definition at line 1100 of file basic_string.h.

4.89.2.19 const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (size_type *__n*) const [inline], [inherited]

Provides access to the data contained in the string.

Parameters

__n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

Definition at line 864 of file basic_string.h.

4.89.2.20 reference std::basic_string< _CharT, _Traits, _Allocator >::at (size_type __n) [inline], [inherited]

Provides access to the data contained in the string.

Parameters

__n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

Definition at line 883 of file basic_string.h.

4.89.2.21 reference std::basic_string< _CharT, _Traits, _Allocator >::back ( ) [inline], [inherited]

Returns a read/write reference to the data at the last element of the string.

Definition at line 913 of file basic_string.h.

4.89.2.22 const_reference std::basic_string< _CharT, _Traits, _Allocator >::back ( ) const [inline], [inherited]

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 921 of file basic_string.h.

4.89.2.23 size_type std::basic_string< _CharT, _Traits, _Allocator >::capacity ( ) const [inline], [noexcept], [inherited]

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 776 of file basic_string.h.

4.89.2.24 int std::basic_string< _CharT, _Traits, _Allocator >::compare (const basic_string< _CharT, _Traits, _Allocator > & __str) const [inline], [inherited]

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2225 of file basic_string.h.

4.89.2.25 int std::basic_string< _CharT, _Traits, _Allocator >::compare (size_type *__pos,* size_type *__n,* const basic_string< _CharT, _Traits, _Allocator > & *__str*) const `[inherited]`

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer $<$ 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or $>$ 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.89.2.26 int std::basic_string< _CharT, _Traits, _Allocator >::compare (size_type *__pos1,* size_type *__n1,* const basic_string< _CharT, _Traits, _Allocator > & *__str,* size_type *__pos2,* size_type *__n2*) const `[inherited]`

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer $<$ 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or $>$ 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

4.89.2.27 bool std::basic_string< _CharT, _Traits, _Allocator >::empty ( ) const `[inline]`, `[noexcept]`, `[inherited]`

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 811 of file basic_string.h.

4.89.2.28 iterator std::basic_string< _CharT, _Traits, _Allocator >::erase (iterator *__position*) [inline], [inherited]

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1378 of file basic_string.h.

4.89.2.29 iterator std::basic_string< _CharT, _Traits, _Allocator >::erase (iterator *__first,* iterator *__last*) [inherited]

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

4.89.2.30 size_type std::basic_string< _CharT, _Traits, _Allocator >::find (const basic_string< _CharT, _Traits, _Allocator > & *__str,* size_type *__pos* = 0) const [inline], [noexcept], [inherited]

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1846 of file basic_string.h.

4.89.2.31 size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (const basic_string< _CharT, _Traits, _Allocator > & *__str,* size_type *__pos* = 0) const [inline], [noexcept], [inherited]

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2079 of file basic_string.h.

4.89.2.32 size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (const basic_string< _CharT, _Traits, _Allocator > & *__str*, size_type *__pos* = 0) const [inline], [noexcept], [inherited]

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1952 of file basic_string.h.

4.89.2.33 size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (const basic_string< _CharT, _Traits, _Allocator > & *__str*, size_type *__pos* = npos) const [inline], [noexcept], [inherited]

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2142 of file basic_string.h.

4.89.2.34 size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (const basic_string< _CharT, _Traits, _Allocator > & *__str*, size_type *__pos* = npos) const [inline], [noexcept], [inherited]

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2016 of file basic_string.h.

4.89.2.35 reference std::basic_string< _CharT, _Traits, _Allocator >::front ( ) [inline], [inherited]

Returns a read/write reference to the data at the first element of the string.

Definition at line 897 of file basic_string.h.

4.89.2.36 const_reference std::basic_string< _CharT, _Traits, _Allocator >::front ( ) const [inline], [inherited]

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 905 of file basic_string.h.

4.89.2.37 allocator_type std::basic_string< _CharT, _Traits, _Allocator >::get_allocator ( ) const [inline], [noexcept], [inherited]

Return copy of allocator used to construct this string.

Definition at line 1817 of file basic_string.h.

4.89.2.38 void std::basic_string< _CharT, _Traits, _Allocator >::insert (iterator *__p*, size_type *__n*, _CharT *__c*) [inline], [inherited]

Insert multiple characters.

Parameters

__p	Iterator referencing location in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1185 of file basic_string.h.

4.89.2.39 void std::basic_string< _CharT, _Traits, _Allocator >::insert (iterator *__p*, _InputIterator *__beg*, _InputIterator *__end*) [inline], [inherited]

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [__beg,__end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1202 of file basic_string.h.

4.89.2.40 void **std::basic_string**< _CharT, _Traits, _Allocator >::insert (iterator *__p*, **initializer_list**< _CharT > *__l*) `[inline]`, `[inherited]`

Insert an initializer_list of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Definition at line 1213 of file basic_string.h.

4.89.2.41 **basic_string**& **std::basic_string**< _CharT, _Traits, _Allocator >::insert (size_type *__pos1*, const **basic_string**< _CharT, _Traits, _Allocator > & *__str*) `[inline]`, `[inherited]`

Insert value of a string.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file basic_string.h.

4.89.2.42 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::insert (size_type __pos1, const basic_string< _CharT, _Traits, _Allocator > & __str, size_type __pos2, size_type __n) `[inline]`, `[inherited]`

Insert a substring.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1255 of file basic_string.h.

4.89.2.43 iterator std::basic_string< _CharT, _Traits, _Allocator >::insert (iterator __p, _CharT __c) `[inline]`, `[inherited]`

Insert one character.

Parameters

__p	Iterator referencing position in string to insert at.
__c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1337 of file basic_string.h.

4.89.2.44 size_type std::basic_string< _CharT, _Traits, _Allocator >::length ( ) const `[inline]`, `[noexcept]`, `[inherited]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 721 of file basic_string.h.

4.89.2.45 size_type std::basic_string< _CharT, _Traits, _Allocator >::max_size ( ) const [inline], [noexcept], [inherited]

Returns the size() of the largest possible string.

Definition at line 726 of file basic_string.h.

4.89.2.46 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::operator+= (const basic_string< _CharT, _Traits, _Allocator > & __str) [inline], [inherited]

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Definition at line 932 of file basic_string.h.

4.89.2.47 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (size_type __pos, size_type __n, const basic_string< _CharT, _Traits, _Allocator > & __str) [inline], [inherited]

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1429 of file basic_string.h.

4.89.2.48 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (size_type __pos1, size_type __n1, const basic_string< _CharT, _Traits, _Allocator > & __str, size_type __pos2, size_type __n2) [inline], [inherited]

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1451 of file basic_string.h.

4.89.2.49 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (iterator *__i1*, iterator *__i2*, const basic_string< _CharT, _Traits, _Allocator > & *__str*) [inline], [inherited]

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file basic_string.h.

4.89.2.50 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (iterator *__i1*, iterator *__i2*, const _CharT * *__s*, size_type *__n*) [inline], [inherited]

Replace range of characters with C substring.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.
__n	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first *__n* characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1557 of file basic_string.h.

4.89.2.51 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (iterator *__i1*, iterator *__i2*, const _CharT * *__s*) [inline], [inherited]

Replace range of characters with C string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1578 of file basic_string.h.

4.89.2.52 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (iterator *__i1*, iterator *__i2*, size_type *__n*, _CharT *__c*) [inline], [inherited]

Replace range of characters with multiple characters.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__n	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, *__n* copies of *__c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1599 of file basic_string.h.

4.89.2.53 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (iterator *__i1,* iterator *__i2,* _InputIterator *__k1,* _InputIterator *__k2*) `[inline]`, `[inherited]`

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1623 of file basic_string.h.

4.89.2.54 basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (iterator *__i1,* iterator *__i2,* initializer_list< _CharT > *__l*) `[inline]`, `[inherited]`

Replace range of characters with initializer_list.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1692 of file basic_string.h.

4.89.2.55 void **std::basic_string**< _CharT, _Traits, _Allocator >::reserve (size_type *__res_arg* = `0`) `[inherited]`

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

4.89.2.56 size_type **std::basic_string**< _CharT, _Traits, _Allocator >::rfind (const **basic_string**< _CharT, _Traits, _Allocator > & *__str*, size_type *__pos* = `npos`) const `[inline]`, `[noexcept]`, `[inherited]`

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1891 of file basic_string.h.

4.89.2.57 size_type **std::basic_string**< _CharT, _Traits, _Allocator >::size ( ) const `[inline]`, `[noexcept]`, `[inherited]`

Returns the number of characters in the string, not including any null-termination.

Definition at line 715 of file basic_string.h.

4.89.2.58 void **std::basic_string**< _CharT, _Traits, _Allocator >::swap (**basic_string**< _CharT, _Traits, _Allocator > & *__s*) `[inherited]`

Swap contents with another string.

Parameters

__s	String to swap with.

Exchanges the contents of this string with that of *__s* in constant time.

4.89.3 Member Data Documentation

4.89.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.89.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.89.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.89.3.4 const size_type std::basic_string< _CharT, _Traits, _Allocator >::npos [static], [inherited]

Value returned by various member functions when they fail.

Definition at line 285 of file basic_string.h.

The documentation for this class was generated from the following file:

- debug/string

4.90 __gnu_parallel::__accumulate_binop_reduct< _BinOp > Struct Template Reference

Public Member Functions

- **__accumulate_binop_reduct** (_BinOp &__b)
- template<typename _Result , typename _Addend >
  _Result **operator()** (const _Result &__x, const _Addend &__y)

Public Attributes

- _BinOp & **__binop**

4.90.1 Detailed Description

template<typename _BinOp>struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >

General reduction, using a binary operator.

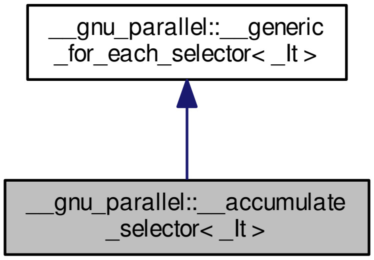
Definition at line 335 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.91 __gnu_parallel::__accumulate_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__accumulate_selector< _It >:



Public Member Functions

- template<typename _Op >
  std::iterator_traits< _It >
  ::value_type operator() (_Op __o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.91.1 Detailed Description

template<typename _It>struct __gnu_parallel::__accumulate_selector< _It >

std::accumulate() selector.

Definition at line 208 of file for_each_selectors.h.

4.91.2 Member Function Documentation

4.91.2.1 template<typename _It> template<typename _Op > std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector< _It >::operator() (_Op *__o*, _It *__i*) [inline]

Functor execution.

Parameters

__o	Operator (unused).
__i	iterator referencing object.

Returns

The current value.

Definition at line 216 of file for_each_selectors.h.

4.91.3 Member Data Documentation

4.91.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

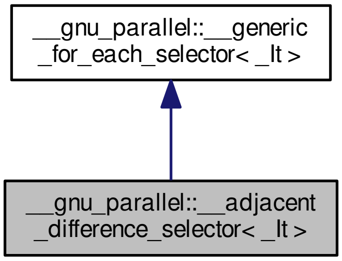
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.92 __gnu_parallel::__adjacent_difference_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__adjacent_difference_selector< _It >:



Public Member Functions

- template<typename _Op >
  bool **operator()** (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.92.1 Detailed Description

template<typename _It>struct __gnu_parallel::__adjacent_difference_selector< _It >

Selector that returns the difference between two adjacent __elements.

Definition at line 269 of file for_each_selectors.h.

4.92.2 Member Data Documentation

4.92.2.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

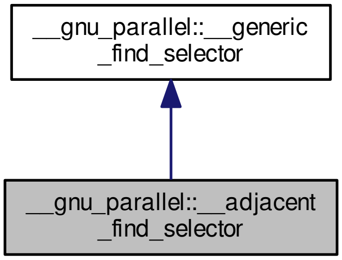
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.93 __gnu_parallel::__adjacent_find_selector Struct Reference

Inheritance diagram for __gnu_parallel::__adjacent_find_selector:



Public Member Functions

- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

4.93.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file find_selectors.h.

4.93.2 Member Function Documentation

4.93.2.1 template<typename _RAIter1 , typename _RAIter2 , typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred*) [inline]

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

Definition at line 105 of file find_selectors.h.

References std::make_pair().

4.93.2.2 template<typename _RAIter1 , typename _RAIter2 , typename _Pred > bool __gnu_parallel::__adjacent_find_selector::operator() (_RAIter1 *__i1,* _RAIter2 *__i2,* _Pred *__pred*) [inline]

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

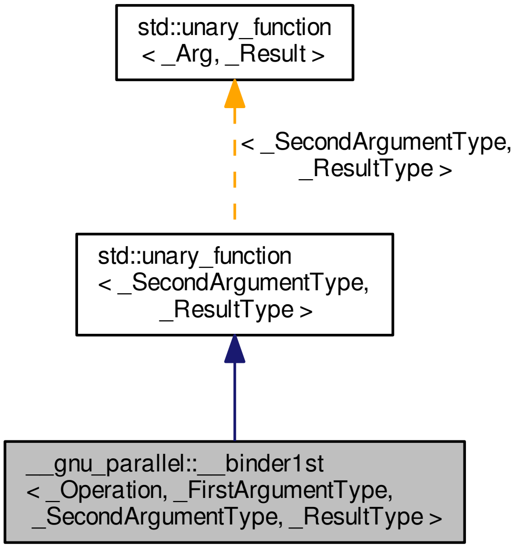
Definition at line 90 of file find_selectors.h.

The documentation for this struct was generated from the following file:

- find_selectors.h

4.94 __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference

Inheritance diagram for __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >:



Public Types

- typedef _SecondArgumentType argument_type
- typedef _ResultType result_type

Public Member Functions

- **__binder1st** (const _Operation &__x, const _FirstArgumentType &__y)
- _ResultType **operator()** (const _SecondArgumentType &__x)
- _ResultType **operator()** (_SecondArgumentType &__x) const

Protected Attributes

- _Operation **_M_op**
- _FirstArgumentType **_M_value**

4.94.1 Detailed Description

template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

Similar to std::binder1st, but giving the argument types explicitly.

Definition at line 192 of file parallel/base.h.

4.94.2 Member Typedef Documentation

4.94.2.1 typedef _SecondArgumentType std::unary_function< _SecondArgumentType , _ResultType >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.94.2.2 typedef _ResultType std::unary_function< _SecondArgumentType , _ResultType >::result_type [inherited]

`result_type` is the return type

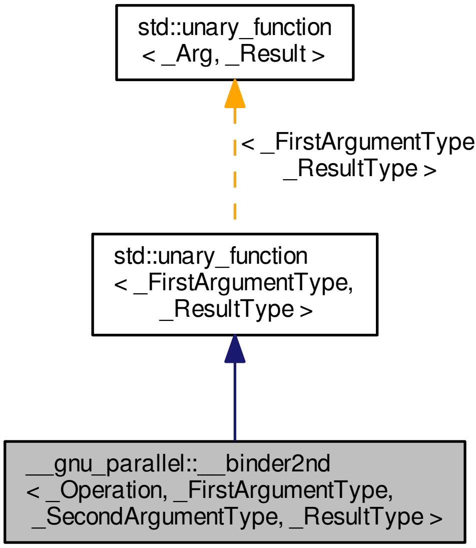
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- parallel/base.h

4.95 __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference

Inheritance diagram for __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >:



Public Types

- typedef _FirstArgumentType argument_type
- typedef _ResultType result_type

Public Member Functions

- **__binder2nd** (const _Operation &__x, const _SecondArgumentType &__y)
- _ResultType **operator()** (const _FirstArgumentType &__x) const
- _ResultType **operator()** (_FirstArgumentType &__x)

Protected Attributes

- _Operation **_M_op**
- _SecondArgumentType **_M_value**

4.95.1 Detailed Description

template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

Similar to std::binder2nd, but giving the argument types explicitly.

Definition at line 220 of file parallel/base.h.

4.95.2 Member Typedef Documentation

4.95.2.1 typedef _FirstArgumentType **std::unary_function**< _FirstArgumentType , _ResultType >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.95.2.2 typedef _ResultType **std::unary_function**< _FirstArgumentType , _ResultType >**::result_type** `[inherited]`

`result_type` is the return type

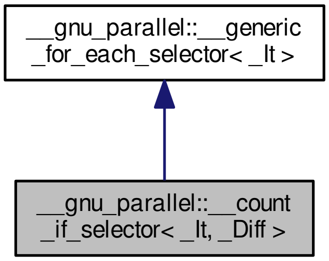
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- parallel/base.h

4.96 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference

Inheritance diagram for __gnu_parallel::__count_if_selector< _It, _Diff >:



Public Member Functions

- template<typename _Op >
  _Diff operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.96.1 Detailed Description

template<typename _It, typename _Diff>struct __gnu_parallel::__count_if_selector< _It, _Diff >

std::count_if () selector.

Definition at line 194 of file for_each_selectors.h.

4.96.2 Member Function Documentation

4.96.2.1 template<typename _It , typename _Diff > template<typename _Op > _Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator() (_Op & __o, _It __i) [inline]

Functor execution.

Parameters

__o	Operator.
__i	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 202 of file for_each_selectors.h.

4.96.3 Member Data Documentation

4.96.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

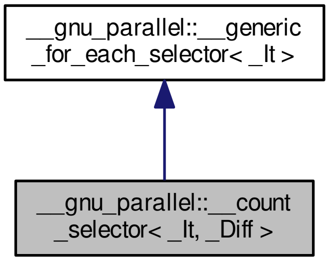
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.97 __gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference

Inheritance diagram for __gnu_parallel::__count_selector< _It, _Diff >:



Public Member Functions

- template<typename _ValueType >
  _Diff operator() (_ValueType &__v, _It __i)

Public Attributes

- _It _M_finish_iterator

4.97.1 Detailed Description

template<typename _It, typename _Diff>struct __gnu_parallel::__count_selector< _It, _Diff >

std::count() selector.

Definition at line 180 of file for_each_selectors.h.

4.97.2 Member Function Documentation

4.97.2.1 template<typename _It, typename _Diff> template<typename _ValueType > _Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (_ValueType & __v, _It __i) [inline]

Functor execution.

Parameters

__v	Current value.
__i	iterator referencing object.

Returns

1 if count, 0 if does not count.

Definition at line 188 of file for_each_selectors.h.

4.97.3 Member Data Documentation

4.97.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

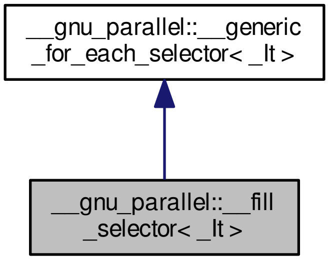
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.98 __gnu_parallel::__fill_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__fill_selector< _It >:



Public Member Functions

- template<typename _ValueType >
  bool operator() (_ValueType &__v, _It __i)

Public Attributes

- _It _M_finish_iterator

4.98.1 Detailed Description

template<typename _It>struct __gnu_parallel::__fill_selector< _It >

std::fill() selector.

Definition at line 84 of file for_each_selectors.h.

4.98.2 Member Function Documentation

4.98.2.1 template<typename _It > template<typename _ValueType > bool __gnu_parallel::__fill_selector< _It >::operator() (_ValueType & __v, _It __i) [inline]

Functor execution.

Parameters

__v	Current value.
__i	iterator referencing object.

Definition at line 91 of file for_each_selectors.h.

4.98.3 Member Data Documentation

4.98.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

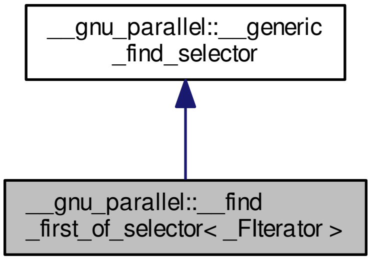
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.99 __gnu_parallel::__find_first_of_selector< _FIterator > Struct Template Reference

Inheritance diagram for __gnu_parallel::__find_first_of_selector< _FIterator >:



Public Member Functions

- **__find_first_of_selector** (_FIterator __begin, _FIterator __end)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

Public Attributes

- _FIterator **_M_begin**
- _FIterator **_M_end**

4.99.1 Detailed Description

template<typename _FIterator>struct __gnu_parallel::__find_first_of_selector< _FIterator >

Test predicate on several elements.

Definition at line 153 of file find_selectors.h.

4.99.2 Member Function Documentation

4.99.2.1 template<typename _FIterator > template<typename _RAIter1 , typename _RAIter2 , typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector< _FIterator >::_M_sequential_algorithm (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred*) [inline]

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

Definition at line 186 of file find_selectors.h.

References std::make_pair().

4.99.2.2 template<typename _FIterator > template<typename _RAIter1 , typename _RAIter2 , typename _Pred > bool __gnu_parallel::__find_first_of_selector< _FIterator >::operator() (_RAIter1 *__i1,* _RAIter2 *__i2,* _Pred *__pred*) [inline]

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

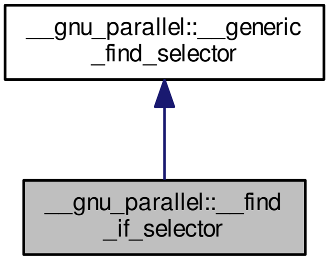
Definition at line 169 of file find_selectors.h.

The documentation for this struct was generated from the following file:

- find_selectors.h

4.100 __gnu_parallel::__find_if_selector Struct Reference

Inheritance diagram for __gnu_parallel::__find_if_selector:



Public Member Functions

- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

4.100.1 Detailed Description

Test predicate on a single element, used for std::find() and std::find_if ().

Definition at line 50 of file find_selectors.h.

4.100.2 Member Function Documentation

4.100.2.1 template<typename _RAIter1 , typename _RAIter2 , typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_if_selector::_M_sequential_algorithm (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred*) [inline]

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

Definition at line 72 of file find_selectors.h.

References std::make_pair().

4.100.2.2 template<typename _RAIter1 , typename _RAIter2 , typename _Pred > bool __gnu_parallel::__find_if_selector::operator() (_RAIter1 *__i1*, _RAIter2 *__i2*, _Pred *__pred*) [inline]

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

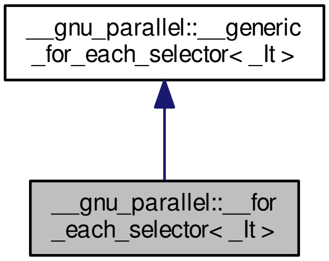
Definition at line 60 of file find_selectors.h.

The documentation for this struct was generated from the following file:

- find_selectors.h

4.101 __gnu_parallel::__for_each_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__for_each_selector< _It >:



Public Member Functions

- template<typename _Op >
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.101.1 Detailed Description

template<typename _It>struct __gnu_parallel::__for_each_selector< _It >

std::for_each() selector.

Definition at line 52 of file for_each_selectors.h.

4.101.2 Member Function Documentation

4.101.2.1 template<typename _It> template<typename _Op > bool __gnu_parallel::__for_each_selector< _It >::operator() (_Op & *__o*, _It *__i*) [inline]

Functor execution.

Parameters

__o	Operator.
__i	iterator referencing object.

Definition at line 59 of file for_each_selectors.h.

4.101.3 Member Data Documentation

4.101.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

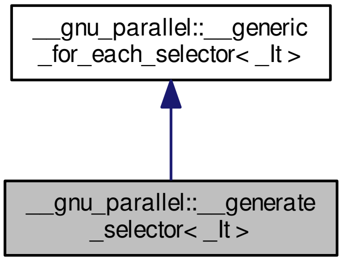
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.102 __gnu_parallel::__generate_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__generate_selector< _It >:



Public Member Functions

- template<typename _Op >
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.102.1 Detailed Description

template<typename _It>struct __gnu_parallel::__generate_selector< _It >

std::generate() selector.

Definition at line 68 of file for_each_selectors.h.

4.102.2 Member Function Documentation

4.102.2.1 template<typename _It> template<typename _Op > bool __gnu_parallel::__generate_selector< _It >::operator() (_Op & __o, _It __i) `[inline]`

Functor execution.

Parameters

__o	Operator.
__i	iterator referencing object.

Definition at line 75 of file for_each_selectors.h.

4.102.3 Member Data Documentation

4.102.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

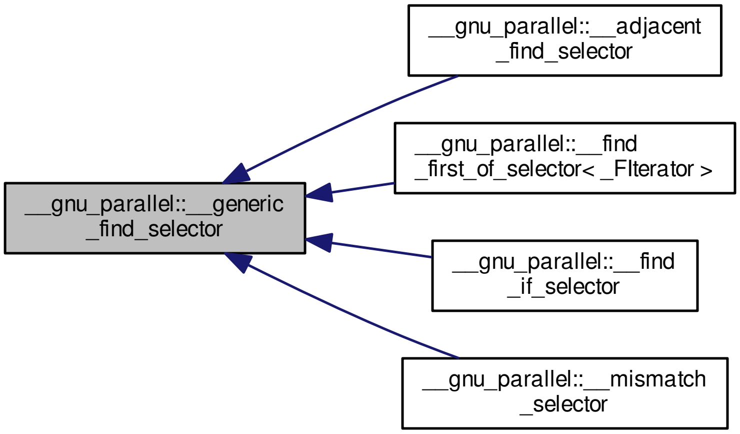
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.103 __gnu_parallel::__generic_find_selector Struct Reference

Inheritance diagram for __gnu_parallel::__generic_find_selector:



4.103.1 Detailed Description

Base class of all __gnu_parallel::__find_template selectors.

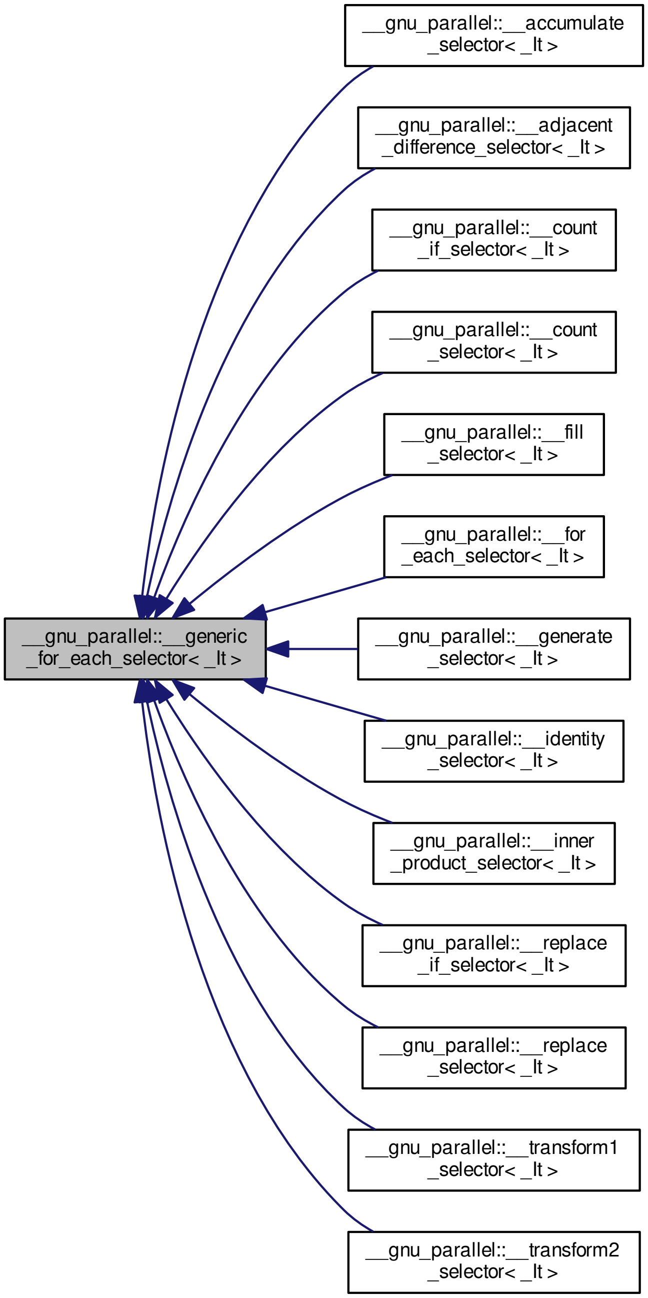
Definition at line 43 of file find_selectors.h.

The documentation for this struct was generated from the following file:

- find_selectors.h

4.104 __gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__generic_for_each_selector< _It >:



Public Attributes

- _It _M_finish_iterator

4.104.1 Detailed Description

template<typename _It>struct __gnu_parallel::__generic_for_each_selector< _It >

Generic __selector for embarrassingly parallel functions.

Definition at line 42 of file for_each_selectors.h.

4.104.2 Member Data Documentation

4.104.2.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

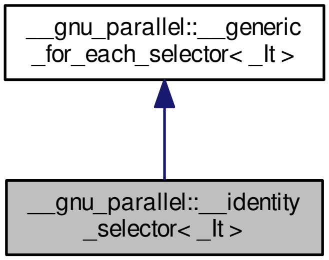
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.105 __gnu_parallel::__identity_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__identity_selector< _It >:



Public Member Functions

- template<typename _Op >
  _It operator() (_Op __o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.105.1 Detailed Description

template<typename _It>struct __gnu_parallel::__identity_selector< _It >

Selector that just returns the passed iterator.

Definition at line 253 of file for_each_selectors.h.

4.105.2 Member Function Documentation

4.105.2.1 template<typename _It> template<typename _Op > _It __gnu_parallel::__identity_selector< _It >::operator() (_Op __o, _It __i) [inline]

Functor execution.

Parameters

__o	Operator (unused).
__i	iterator referencing object.

Returns

Passed iterator.

Definition at line 261 of file for_each_selectors.h.

4.105.3 Member Data Documentation

4.105.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

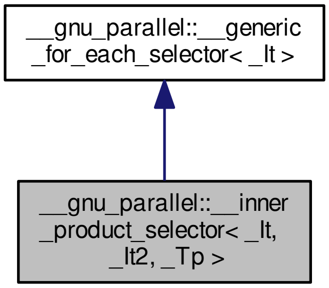
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.106 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference

Inheritance diagram for __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >:



Public Member Functions

- __inner_product_selector (_It __b1, _It2 __b2)
- template<typename _Op >
  _Tp operator() (_Op __mult, _It __current)

Public Attributes

- _It __begin1_iterator
- _It2 __begin2_iterator
- _It _M_finish_iterator

4.106.1 Detailed Description

template<typename _It, typename _It2, typename _Tp>struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >

std::inner_product() selector.

Definition at line 222 of file for_each_selectors.h.

4.106.2 Constructor & Destructor Documentation

4.106.2.1 template<typename _It , typename _It2 , typename _Tp > __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__inner_product_selector (_It *__b1,* _It2 *__b2*) [inline], [explicit]

Constructor.

Parameters

__b1	Begin iterator of first sequence.
__b2	Begin iterator of second sequence.

Definition at line 234 of file for_each_selectors.h.

4.106.3 Member Function Documentation

4.106.3.1 template<typename _It , typename _It2 , typename _Tp > template<typename _Op > _Tp **__gnu_parallel::__inner_product_selector**< _It, _It2, _Tp >::operator() (_Op *__mult*, _It *__current*) [inline]

Functor execution.

Parameters

__mult	Multiplication functor.
__current	iterator referencing object.

Returns

Inner product elemental __result.

Definition at line 243 of file for_each_selectors.h.

References __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator, and __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator.

4.106.4 Member Data Documentation

4.106.4.1 template<typename _It , typename _It2 , typename _Tp > _It **__gnu_parallel::__inner_product_selector**< _It, _It2, _Tp >::__begin1_iterator

Begin iterator of first sequence.

Definition at line 225 of file for_each_selectors.h.

Referenced by __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator()().

4.106.4.2 template<typename _It , typename _It2 , typename _Tp > _It2 **__gnu_parallel::__inner_product_selector**< _It, _It2, _Tp >::__begin2_iterator

Begin iterator of second sequence.

Definition at line 228 of file for_each_selectors.h.

Referenced by __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator()().

4.106.4.3 template<typename _It > _It **__gnu_parallel::__generic_for_each_selector**< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.107 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference

Public Member Functions

- **__max_element_reduct** (_Compare &__c)
- _It **operator()** (_It __x, _It __y)

Public Attributes

- _Compare & **__comp**

4.107.1 Detailed Description

template<typename _Compare, typename _It>struct __gnu_parallel::__max_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.108 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference

Public Member Functions

- **__min_element_reduct** (_Compare &__c)
- _It **operator()** (_It __x, _It __y)

Public Attributes

- _Compare & **__comp**

4.108.1 Detailed Description

template<typename _Compare, typename _It>struct __gnu_parallel::__min_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.

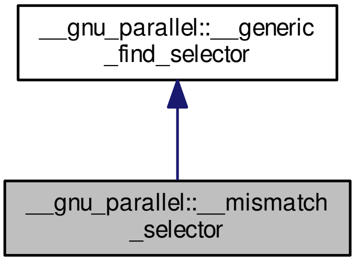
Definition at line 307 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.109 __gnu_parallel::__mismatch_selector Struct Reference

Inheritance diagram for __gnu_parallel::__mismatch_selector:



Public Member Functions

- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

4.109.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file find_selectors.h.

4.109.2 Member Function Documentation

4.109.2.1 template<typename _RAIter1 , typename _RAIter2 , typename _Pred > std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (_RAIter1 *__begin1,* _RAIter1 *__end1,* _RAIter2 *__begin2,* _Pred *__pred*) [inline]

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

Definition at line 143 of file find_selectors.h.

4.109.2.2 template<typename _RAIter1 , typename _RAIter2 , typename _Pred > bool __gnu_parallel::__mismatch_selector::operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred) [inline]

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

Definition at line 130 of file find_selectors.h.

The documentation for this struct was generated from the following file:

- find_selectors.h

4.110 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)

4.110.1 Detailed Description

template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned off.

Note that 3-way merging is always stable!

Definition at line 752 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.111 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)

4.111.1 Detailed Description

template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned on.

Note that 3-way merging is always stable!

Definition at line 772 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.112 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)

4.112.1 Detailed Description

template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 4-way merging with __sentinels turned off.

Note that 4-way merging is always stable!

Definition at line 795 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.113 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)

4.113.1 Detailed Description

template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 4-way merging with __sentinels turned on.

Note that 4-way merging is always stable!

Definition at line 815 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.114 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)

4.114.1 Detailed Description

template<bool __sentinels, bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for k-way merging with __sentinels turned on.

Definition at line 837 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.115 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)

4.115.1 Detailed Description

template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for k-way merging with __sentinels turned off.

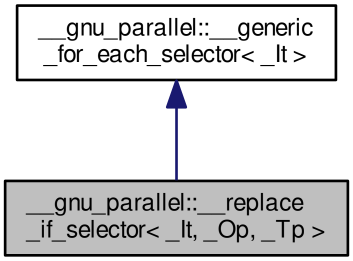
Definition at line 872 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.116 __gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Template Reference

Inheritance diagram for __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >:



Public Member Functions

- __replace_if_selector (const _Tp &__new_val)
- bool operator() (_Op &__o, _It __i)

Public Attributes

- const _Tp & __new_val
- _It _M_finish_iterator

4.116.1 Detailed Description

template<typename _It, typename _Op, typename _Tp>struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >

std::replace() selector.

Definition at line 156 of file for_each_selectors.h.

4.116.2 Constructor & Destructor Documentation

4.116.2.1 template<typename _It , typename _Op , typename _Tp > __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (const _Tp & *__new_val*) [inline], [explicit]

Constructor.

Parameters

__new_val	Value to replace with.

Definition at line 164 of file for_each_selectors.h.

4.116.3 Member Function Documentation

4.116.3.1 template<typename _It , typename _Op , typename _Tp > bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (_Op & __o, _It __i) [inline]

Functor execution.

Parameters

__o	Operator.
__i	iterator referencing object.

Definition at line 170 of file for_each_selectors.h.

References __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val.

4.116.4 Member Data Documentation

4.116.4.1 template<typename _It , typename _Op , typename _Tp > const _Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val

Value to replace with.

Definition at line 159 of file for_each_selectors.h.

Referenced by __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator()().

4.116.4.2 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

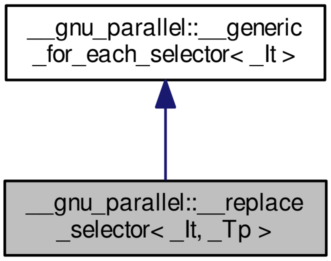
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.117 __gnu_parallel::__replace_selector< _It, _Tp > Struct Template Reference

Inheritance diagram for __gnu_parallel::__replace_selector< _It, _Tp >:



Public Member Functions

- __replace_selector (const _Tp &__new_val)
- bool operator() (_Tp &__v, _It __i)

Public Attributes

- const _Tp & __new_val
- _It _M_finish_iterator

4.117.1 Detailed Description

template<typename _It, typename _Tp>struct __gnu_parallel::__replace_selector< _It, _Tp >

std::replace() selector.

Definition at line 132 of file for_each_selectors.h.

4.117.2 Constructor & Destructor Documentation

4.117.2.1 template<typename _It , typename _Tp > __gnu_parallel::__replace_selector< _It, _Tp >::__replace_selector (const _Tp & *__new_val*) [inline], [explicit]

Constructor.

Parameters

__new_val	Value to replace with.

Definition at line 140 of file for_each_selectors.h.

4.117.3 Member Function Documentation

4.117.3.1 template<typename _It , typename _Tp > bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (_Tp & __v, _It __i) [inline]

Functor execution.

Parameters

__v	Current value.
__i	iterator referencing object.

Definition at line 146 of file for_each_selectors.h.

References __gnu_parallel::__replace_selector< _It, _Tp >::__new_val.

4.117.4 Member Data Documentation

4.117.4.1 template<typename _It , typename _Tp > const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val

Value to replace with.

Definition at line 135 of file for_each_selectors.h.

Referenced by __gnu_parallel::__replace_selector< _It, _Tp >::operator()().

4.117.4.2 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

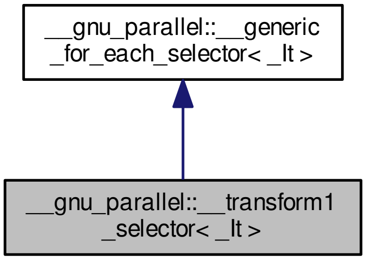
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.118 __gnu_parallel::__transform1_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__transform1_selector< _It >:



Public Member Functions

- template<typename _Op >
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.118.1 Detailed Description

template<typename _It>struct __gnu_parallel::__transform1_selector< _It >

std::transform() __selector, one input sequence variant.

Definition at line 100 of file for_each_selectors.h.

4.118.2 Member Function Documentation

4.118.2.1 template<typename _It> template<typename _Op > bool __gnu_parallel::__transform1_selector< _It >::operator() (_Op & *__o*, _It *__i*) `[inline]`

Functor execution.

Parameters

__o	Operator.
__i	iterator referencing object.

Definition at line 107 of file for_each_selectors.h.

4.118.3 Member Data Documentation

4.118.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator `[inherited]`

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

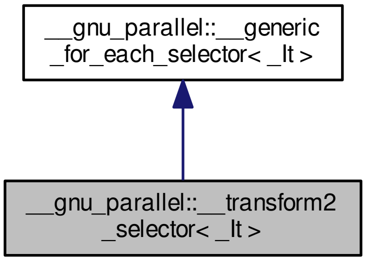
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.119 __gnu_parallel::__transform2_selector< _It > Struct Template Reference

Inheritance diagram for __gnu_parallel::__transform2_selector< _It >:



Public Member Functions

- template<typename _Op >
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

4.119.1 Detailed Description

template<typename _It>struct __gnu_parallel::__transform2_selector< _It >

std::transform() __selector, two input sequences variant.

Definition at line 116 of file for_each_selectors.h.

4.119.2 Member Function Documentation

4.119.2.1 template<typename _It> template<typename _Op > bool __gnu_parallel::__transform2_selector< _It >::operator() (_Op & *__o*, _It *__i*) `[inline]`

Functor execution.

Parameters

__o	Operator.
__i	iterator referencing object.

Definition at line 123 of file for_each_selectors.h.

4.119.3 Member Data Documentation

4.119.3.1 template<typename _It > _It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator `[inherited]`

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).

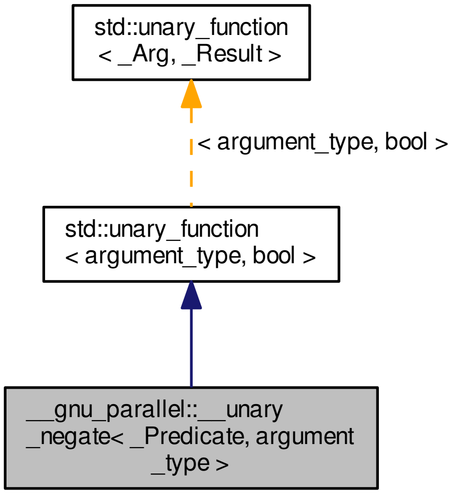
Definition at line 47 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.120 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference

Inheritance diagram for __gnu_parallel::__unary_negate< _Predicate, argument_type >:



Public Types

- typedef argument_type argument_type
- typedef bool result_type

Public Member Functions

- **__unary_negate** (const _Predicate &__x)
- bool **operator()** (const argument_type &__x)

Protected Attributes

- _Predicate **_M_pred**

4.120.1 Detailed Description

template<typename _Predicate, typename argument_type>class __gnu_parallel::__unary_negate< _Predicate, argument_type >

Similar to std::unary_negate, but giving the argument types explicitly.

Definition at line 173 of file parallel/base.h.

4.120.2 Member Typedef Documentation

4.120.2.1 typedef **argument_type std::unary_function< argument_type , bool >::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.120.2.2 typedef bool **std::unary_function< argument_type , bool >::result_type** `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- parallel/base.h

4.121 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::iterator_traits < _RAIter > **_TraitsType**
- typedef _TraitsType::value_type **_ValueType**

Public Member Functions

- _DRandomShufflingGlobalData (_RAIter &__source)

Public Attributes

- _ThreadIndex ∗ _M_bin_proc
- _DifferenceType ∗∗ _M_dist
- int _M_num_bins
- int _M_num_bits
- _RAIter & _M_source
- _DifferenceType ∗ _M_starts
- _ValueType ∗∗ _M_temporaries

4.121.1 Detailed Description

template<typename _RAIter>struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >

Data known to every thread participating in __gnu_parallel::__parallel_random_shuffle().

Definition at line 52 of file random_shuffle.h.

4.121.2 Constructor & Destructor Documentation

4.121.2.1 template<typename _RAIter> __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_DRandomShufflingGlobalData (_RAIter & *__source*) [inline]

Constructor.

Definition at line 83 of file random_shuffle.h.

4.121.3 Member Data Documentation

4.121.3.1 template<typename _RAIter> _ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bin_proc

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

4.121.3.2 template<typename _RAIter> _DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist

Two-dimensional array to hold the thread-bin distribution.

Dimensions (_M_num_threads + 1) __x (_M_num_bins + 1).

Definition at line 67 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.121.3.3 template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins

Number of bins to distribute to.

Definition at line 77 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.121.3.4 template<typename _RAIter> int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits

Number of bits needed to address the bins.

Definition at line 80 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.121.3.5 template<typename _RAIter> _RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source

Begin iterator of the __source.

Definition at line 59 of file random_shuffle.h.

4.121.3.6 template<typename _RAIter> _DifferenceType∗ __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts

Start indexes of the threads' __chunks.

Definition at line 70 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.121.3.7 template<typename _RAIter> _ValueType∗∗ __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries

Temporary arrays for each thread.

Definition at line 62 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

The documentation for this struct was generated from the following file:

- random_shuffle.h

4.122 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference

Public Attributes

- _BinIndex __bins_end
- _BinIndex _M_bins_begin
- int _M_num_threads
- _DRandomShufflingGlobalData
  < _RAIter > ∗ _M_sd
- uint32_t _M_seed

4.122.1 Detailed Description

template<typename _RAIter, typename _RandomNumberGenerator>struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Local data for a thread participating in __gnu_parallel::__parallel_random_shuffle().

Definition at line 91 of file random_shuffle.h.

4.122.2 Member Data Documentation

4.122.2.1 template<typename _RAIter, typename _RandomNumberGenerator> _BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end

End index for bins taken care of by this thread.

Definition at line 100 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

4.122.2.2 template<typename _RAIter, typename _RandomNumberGenerator> **_BinIndex __gnu_parallel::_DRSSorterPU**< _RAIter, _RandomNumberGenerator >::_M_bins_begin

Begin index for bins taken care of by this thread.

Definition at line 97 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

4.122.2.3 template<typename _RAIter, typename _RandomNumberGenerator> int **__gnu_parallel::_DRSSorterPU**< _RAIter, _RandomNumberGenerator >::_M_num_threads

Number of threads participating in total.

Definition at line 94 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.122.2.4 template<typename _RAIter, typename _RandomNumberGenerator> **_DRandomShufflingGlobalData**<_RAIter>* **__gnu_parallel::_DRSSorterPU**< _RAIter, _RandomNumberGenerator >::_M_sd

Pointer to global data.

Definition at line 106 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

4.122.2.5 template<typename _RAIter, typename _RandomNumberGenerator> uint32_t **__gnu_parallel::_DRSSorterPU**< _RAIter, _RandomNumberGenerator >::_M_seed

Random _M_seed for this thread.

Definition at line 103 of file random_shuffle.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

The documentation for this struct was generated from the following file:

- random_shuffle.h

4.123 __gnu_parallel::_DummyReduct Struct Reference

Public Member Functions

- bool **operator()** (bool, bool) const

4.123.1 Detailed Description

Reduction function doing nothing.

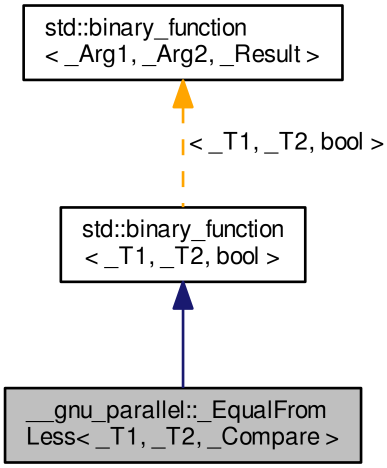
Definition at line 298 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.124 __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- **_EqualFromLess** (_Compare &__comp)
- bool **operator()** (const _T1 &__a, const _T2 &__b)

4.124.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file parallel/base.h.

4.124.2 Member Typedef Documentation

4.124.2.1 typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.124.2.2 typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.124.2.3 typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

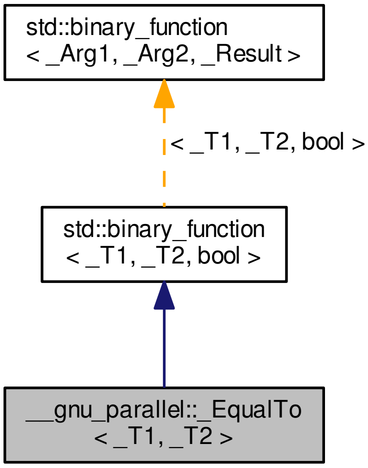
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- parallel/base.h

4.125 __gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference

Inheritance diagram for __gnu_parallel::_EqualTo< _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- bool **operator()** (const _T1 &__t1, const _T2 &__t2) const

4.125.1 Detailed Description

template<typename _T1, typename _T2>struct __gnu_parallel::_EqualTo< _T1, _T2 >

Similar to std::equal_to, but allows two different types.

Definition at line 244 of file parallel/base.h.

4.125.2 Member Typedef Documentation

4.125.2.1 typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.125.2.2 typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.125.2.3 typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- parallel/base.h

4.126 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference

Public Member Functions

- _GuardedIterator (_RAIter __begin, _RAIter __end, _Compare &__comp)
- operator _RAIter ()
- std::iterator_traits< _RAIter >::value_type & operator* ()
- _GuardedIterator< _RAIter, _Compare > & operator++ ()

Friends

- bool operator< (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)
- bool operator<= (_GuardedIterator< _RAIter, _Compare > &__bi1, _GuardedIterator< _RAIter, _Compare > &__bi2)

4.126.1 Detailed Description

template<typename _RAIter, typename _Compare>class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from _RAIter is not possible since _RAIter need not be a class.

Definition at line 73 of file multiway_merge.h.

4.126.2 Constructor & Destructor Documentation

4.126.2.1 template<typename _RAIter , typename _Compare > __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::_GuardedIterator (_RAIter *__begin*, _RAIter *__end*, _Compare & *__comp*) [inline]

Constructor. Sets iterator to beginning of sequence.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator provided for associated overloaded compare operators.

Definition at line 91 of file multiway_merge.h.

4.126.3 Member Function Documentation

4.126.3.1 template<typename _RAIter , typename _Compare > __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter ( ) [inline]

Convert to wrapped iterator.

Returns

Wrapped iterator.

Definition at line 112 of file multiway_merge.h.

4.126.3.2 template<typename _RAIter , typename _Compare > std::iterator_traits<_RAIter>::value_type& __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator* ( ) [inline]

Dereference operator.

Returns

Referenced element.

Definition at line 107 of file multiway_merge.h.

4.126.3.3 template<typename _RAIter , typename _Compare > _GuardedIterator<_RAIter, _Compare>& __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator++ ( ) [inline]

Pre-increment operator.

Returns

This.

Definition at line 98 of file multiway_merge.h.

4.126.4 Friends And Related Function Documentation

4.126.4.1 template<typename _RAIter , typename _Compare > bool operator< (**_GuardedIterator**< _RAIter, _Compare > & *__bi1,* **_GuardedIterator**< _RAIter, _Compare > & *__bi2*) `[friend]`

Compare two elements referenced by guarded iterators.

Parameters

__bi1	First iterator.
__bi2	Second iterator.

Returns

`true` if less.

Definition at line 120 of file multiway_merge.h.

4.126.4.2 template<typename _RAIter , typename _Compare > bool operator<= (**_GuardedIterator**< _RAIter, _Compare > & *__bi1,* **_GuardedIterator**< _RAIter, _Compare > & *__bi2*) `[friend]`

Compare two elements referenced by guarded iterators.

Parameters

__bi1	First iterator.
__bi2	Second iterator.

Returns

`True` if less equal.

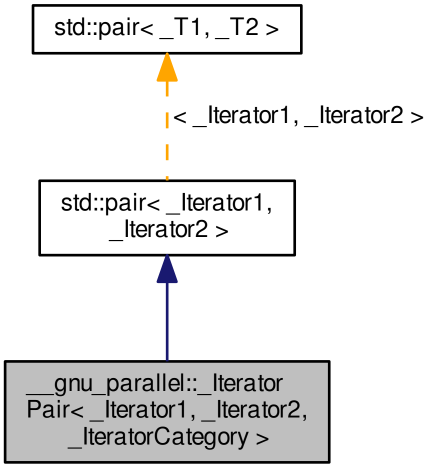
Definition at line 135 of file multiway_merge.h.

The documentation for this class was generated from the following file:

- multiway_merge.h

4.127 __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference

Inheritance diagram for __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >:



Public Types

- typedef std::iterator_traits < _Iterator1 > **_TraitsType**
- typedef _TraitsType::difference_type **difference_type**
- typedef _Iterator1 **first_type**
- typedef _IteratorCategory **iterator_category**
- typedef _IteratorPair ∗ **pointer**
- typedef _IteratorPair & **reference**
- typedef _Iterator2 second_type
- typedef void **value_type**

Public Member Functions

- **_IteratorPair** (const _Iterator1 &__first, const _Iterator2 &__second)
- void __p first && **noexcept** (swap(second, __p.second)))
- **operator _Iterator2** () const
- _IteratorPair **operator+** (difference_type __delta) const
- _IteratorPair & **operator++** ()
- const _IteratorPair **operator++** (int)
- difference_type **operator-** (const _IteratorPair &__other) const
- _IteratorPair & **operator--** ()
- const _IteratorPair **operator--** (int)
- _IteratorPair & **operator=** (const _IteratorPair &__other)
- void **swap** (pair &__p) noexcept(noexcept(swap(first

Public Attributes

- _Iterator1 first
- pair
  is_nothrow_move_assignable
  < _Iterator2 >::value **first**
- _Iterator2 second
- **second**
- return ∗ **this**

4.127.1 Detailed Description

template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file iterator.h.

4.127.2 Member Typedef Documentation

4.127.2.1 typedef _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second_type [inherited]

`first_type` is the first bound type

Definition at line 99 of file stl_pair.h.

4.127.3 Member Data Documentation

4.127.3.1 _Iterator1 std::pair< _Iterator1 , _Iterator2 >::first [inherited]

`second_type` is the second bound type

Definition at line 101 of file stl_pair.h.

4.127.3.2 _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second [inherited]

`first` is a copy of the first object

Definition at line 102 of file stl_pair.h.

The documentation for this class was generated from the following file:

- iterator.h

4.128 __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference

Public Types

- typedef std::iterator_traits
  < _Iterator1 >
  ::difference_type **difference_type**

- typedef _IteratorCategory **iterator_category**
- typedef _IteratorTriple ∗ **pointer**
- typedef _IteratorTriple & **reference**
- typedef void **value_type**

Public Member Functions

- **_IteratorTriple** (const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third)
- **operator _Iterator3** () const
- _IteratorTriple **operator+** (difference_type __delta) const
- _IteratorTriple & **operator++** ()
- const _IteratorTriple **operator++** (int)
- difference_type **operator-** (const _IteratorTriple &__other) const
- _IteratorTriple & **operator--** ()
- const _IteratorTriple **operator--** (int)
- _IteratorTriple & **operator=** (const _IteratorTriple &__other)

Public Attributes

- _Iterator1 **_M_first**
- _Iterator2 **_M_second**
- _Iterator3 **_M_third**

4.128.1 Detailed Description

template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file iterator.h.

The documentation for this class was generated from the following file:

- iterator.h

4.129 __gnu_parallel::_Job< _DifferenceTp > Struct Template Reference

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Attributes

- volatile _DifferenceType _M_first
- volatile _DifferenceType _M_last
- volatile _DifferenceType _M_load

4.129.1 Detailed Description

template<typename _DifferenceTp>struct __gnu_parallel::_Job< _DifferenceTp >

One __job for a certain thread.

Definition at line 54 of file workstealing.h.

4.129.2 Member Data Documentation

4.129.2.1 template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_first

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file workstealing.h.

Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

4.129.2.2 template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_last

Last element.

Changed by owning thread only.

Definition at line 67 of file workstealing.h.

Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

4.129.2.3 template<typename _DifferenceTp> volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load

Number of elements, i.e. `_M_last-_M_first+1`.

Changed by owning thread only.

Definition at line 72 of file workstealing.h.

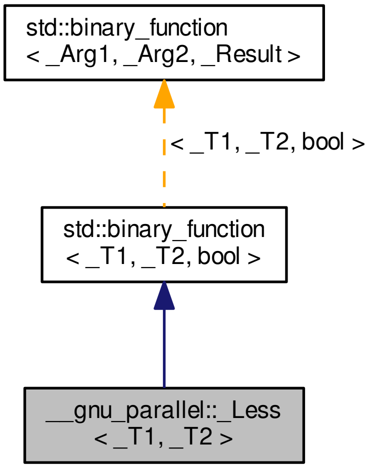
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

The documentation for this struct was generated from the following file:

- workstealing.h

4.130 __gnu_parallel::_Less< _T1, _T2 > Struct Template Reference

Inheritance diagram for __gnu_parallel::_Less< _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- bool **operator()** (const _T1 &__t1, const _T2 &__t2) const
- bool **operator()** (const _T2 &__t2, const _T1 &__t1) const

4.130.1 Detailed Description

template<typename _T1, typename _T2>struct __gnu_parallel::_Less< _T1, _T2 >

Similar to std::less, but allows two different types.

Definition at line 252 of file parallel/base.h.

4.130.2 Member Typedef Documentation

4.130.2.1 typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.130.2.2 typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.130.2.3 typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

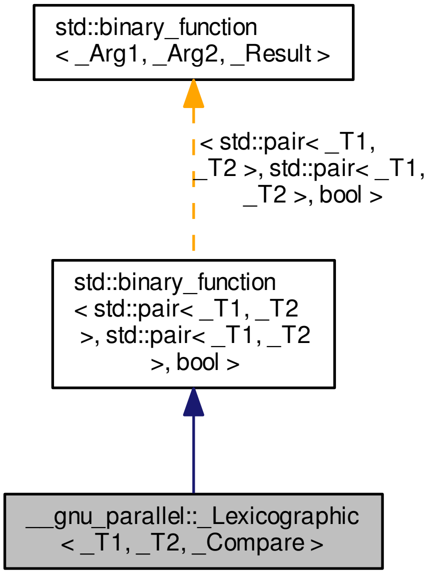
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- parallel/base.h

4.131 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >:



Public Types

- typedef std::pair< _T1, _T2 > first_argument_type
- typedef bool result_type
- typedef std::pair< _T1, _T2 > second_argument_type

Public Member Functions

- **_Lexicographic** (_Compare &__comp)
- bool **operator()** (const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2) const

4.131.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >

Compare __a pair of types lexicographically, ascending.

Definition at line 53 of file multiseq_selection.h.

4.131.2 Member Typedef Documentation

4.131.2.1 typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.131.2.2 typedef bool std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.131.2.3 typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

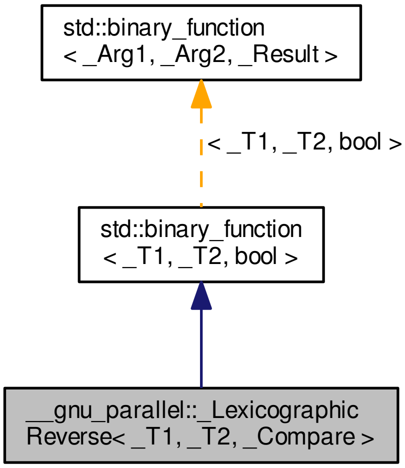
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- multiseq_selection.h

4.132 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- **_LexicographicReverse** (_Compare &__comp)
- bool **operator()** (const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2) const

4.132.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >

Compare __a pair of types lexicographically, descending.

Definition at line 80 of file multiseq_selection.h.

4.132.2 Member Typedef Documentation

4.132.2.1 typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.132.2.2 typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.132.2.3 typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

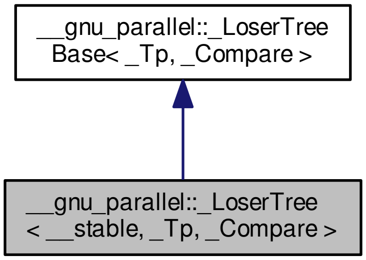
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- multiseq_selection.h

4.133 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)
- void __delete_min_insert (_Tp __key, bool __sup)
- int __get_min_source ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare _M_comp
- bool _M_first_insert
- unsigned int **_M_ik**
- unsigned int **_M_k**
- unsigned int _M_log_k
- _Loser $*$ _M_losers
- unsigned int **_M_offset**

4.133.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >

Stable _LoserTree variant.

Provides the stable implementations of insert_start, __init_winner, __init and __delete_min_insert.

Unstable variant is done using partial specialisation below.

Definition at line 169 of file losertree.h.

4.133.2 Member Function Documentation

4.133.2.1 template<bool __stable, typename _Tp , typename _Compare > void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (_Tp *__key*, bool *__sup*) `[inline]`

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 222 of file losertree.h.

4.133.2.2 template<typename _Tp , typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ( ) `[inline]`, `[inherited]`

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source.

4.133.2.3 template<typename _Tp , typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (const _Tp & *__key*, int *__source*, bool *__sup*) `[inline]`, `[inherited]`

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

Definition at line 134 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup.

4.133.3 Member Data Documentation

4.133.3.1 template<typename _Tp , typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected], [inherited]

_Compare to use.

Definition at line 78 of file losertree.h.

4.133.3.2 template<typename _Tp , typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected], [inherited]

State flag that determines whether the _LoserTree is empty.

Only used for building the _LoserTree.

Definition at line 85 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase().

4.133.3.3 template<typename _Tp , typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]

log_2{_M_k}

Definition at line 72 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase().

4.133.3.4 template<typename _Tp , typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers [protected], [inherited]

_LoserTree __elements.

Definition at line 75 of file losertree.h.

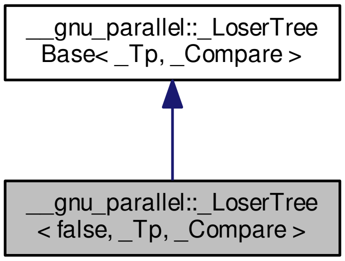
Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::∼_LoserTreeBase().

The documentation for this class was generated from the following file:

- losertree.h

4.134 __gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTree< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)
- void __delete_min_insert (_Tp __key, bool __sup)
- int __get_min_source ()
- void **__init** ()
- unsigned int __init_winner (unsigned int __root)
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare _M_comp
- bool _M_first_insert
- unsigned int **_M_ik**
- unsigned int **_M_k**
- unsigned int _M_log_k
- _Loser ∗ _M_losers
- unsigned int **_M_offset**

4.134.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTree< false, _Tp, _Compare >

Unstable _LoserTree variant.

Stability (non-stable here) is selected with partial specialization.

Definition at line 261 of file losertree.h.

4.134.2 Member Function Documentation

4.134.2.1 template<typename _Tp , typename _Compare > void **__gnu_parallel::_LoserTree**< false, _Tp, _Compare >::__delete_min_insert (_Tp *__key,* bool *__sup*) [inline]

Delete the _M_key smallest element and insert the element __key instead.

Parameters

__key	the _M_key to insert
__sup	true iff __key is an explicitly marked supremum

Definition at line 324 of file losertree.h.

4.134.2.2 template<typename _Tp , typename _Compare > int **__gnu_parallel::_LoserTreeBase**< _Tp, _Compare >::__get_min_source ( ) [inline], [inherited]

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source.

4.134.2.3 template<typename _Tp , typename _Compare > unsigned int **__gnu_parallel::_LoserTree**< false, _Tp, _Compare >::__init_winner (unsigned int *__root*) [inline]

Computes the winner of the competition at position "__root".

Called recursively (starting at 0) to build the initial tree.

Parameters

__root	__index of the "game" to start.

Definition at line 284 of file losertree.h.

4.134.2.4 template<typename _Tp , typename _Compare > void **__gnu_parallel::_LoserTreeBase**< _Tp, _Compare >::__insert_start (const _Tp & *__key,* int *__source,* bool *__sup*) [inline], [inherited]

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

Definition at line 134 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup.

4.134.3 Member Data Documentation

4.134.3.1 template<typename _Tp , typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected], [inherited]

_Compare to use.

Definition at line 78 of file losertree.h.

4.134.3.2 template<typename _Tp , typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected], [inherited]

State flag that determines whether the _LoserTree is empty.

Only used for building the _LoserTree.

Definition at line 85 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase().

4.134.3.3 template<typename _Tp , typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]

log_2{_M_k}

Definition at line 72 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase().

4.134.3.4 template<typename _Tp , typename _Compare > _Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers [protected], [inherited]

_LoserTree __elements.

Definition at line 75 of file losertree.h.

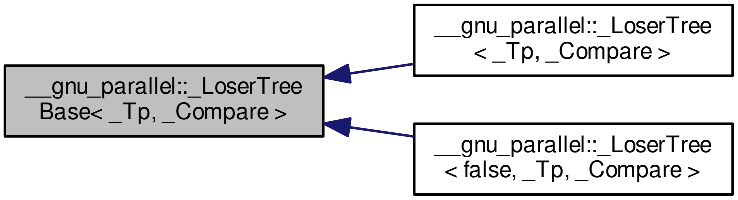
Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~_LoserTreeBase().

The documentation for this class was generated from the following file:

- losertree.h

4.135 __gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreeBase< _Tp, _Compare >:



Classes

- struct _Loser

  Internal representation of a _LoserTree element.

Public Member Functions

- _LoserTreeBase (unsigned int __k, _Compare __comp)
- ∼_LoserTreeBase ()
- int __get_min_source ()
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare _M_comp
- bool _M_first_insert
- unsigned int **_M_ik**
- unsigned int **_M_k**
- unsigned int _M_log_k
- _Loser ∗ _M_losers
- unsigned int **_M_offset**

4.135.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag _M_sup per element, inf is not needed due to a better initialization routine. This is a well-performing variant.

Parameters

_Tp	the element type
_Compare	the comparator to use, defaults to std::less<_Tp>

Definition at line 55 of file losertree.h.

4.135.2 Constructor & Destructor Documentation

4.135.2.1 template<typename _Tp , typename _Compare > __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (unsigned int *__k,* _Compare *__comp*) [inline]

The constructor.

Parameters

__k	The number of sequences to merge.
__comp	The comparator to use.

Definition at line 94 of file losertree.h.

References __gnu_parallel::__rd_log2(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers.

4.135.2.2 template<typename _Tp , typename _Compare > __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::∼_LoserTreeBase ( ) [inline]

The destructor.

Definition at line 118 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers.

4.135.3 Member Function Documentation

4.135.3.1 template<typename _Tp , typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ( ) [inline]

Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source.

4.135.3.2 template<typename _Tp , typename _Compare > void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (const _Tp & *__key,* int *__source,* bool *__sup*) [inline]

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

Definition at line 134 of file losertree.h.

References __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers, __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source, and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup.

4.135.4 Member Data Documentation

4.135.4.1 template<typename _Tp , typename _Compare > _Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp `[protected]`

_Compare to use.

Definition at line 78 of file losertree.h.

4.135.4.2 template<typename _Tp , typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert `[protected]`

State flag that determines whether the _LoserTree is empty.

Only used for building the _LoserTree.

Definition at line 85 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase().

4.135.4.3 template<typename _Tp , typename _Compare > unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k `[protected]`

log_2{_M_k}

Definition at line 72 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase().

4.135.4.4 template<typename _Tp , typename _Compare > _Loser∗ __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers `[protected]`

_LoserTree __elements.

Definition at line 75 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start(), __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::∼_LoserTreeBase().

The documentation for this class was generated from the following file:

- losertree.h

4.136 __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference

Public Attributes

- _Tp _M_key
- int _M_source
- bool _M_sup

4.136.1 Detailed Description

template<typename _Tp, typename _Compare>struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser

Internal representation of a _LoserTree element.

Definition at line 59 of file losertree.h.

4.136.2 Member Data Documentation

4.136.2.1 template<typename _Tp , typename _Compare > _Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key

_M_key of the element in the _LoserTree.

Definition at line 66 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start().

4.136.2.2 template<typename _Tp , typename _Compare > int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source

__index of the __source __sequence.

Definition at line 64 of file losertree.h.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source(), and __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start().

4.136.2.3 template<typename _Tp , typename _Compare > bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup

flag, true iff this is a "maximum" __sentinel.

Definition at line 62 of file losertree.h.

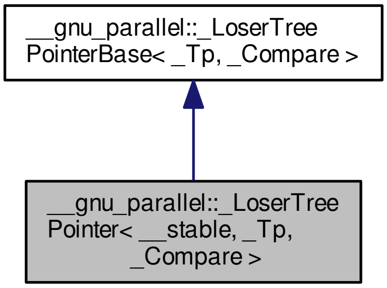
Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start().

The documentation for this struct was generated from the following file:

- losertree.h

4.137 __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.137.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >

Stable _LoserTree implementation.

The unstable variant is implemented using partial instantiation below.

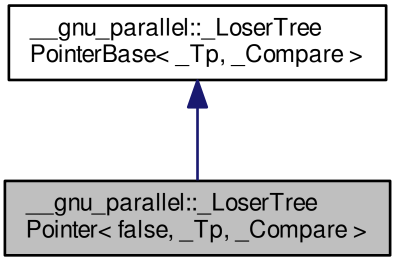
Definition at line 409 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.138 __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.138.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >

Unstable _LoserTree implementation.

The stable variant is above.

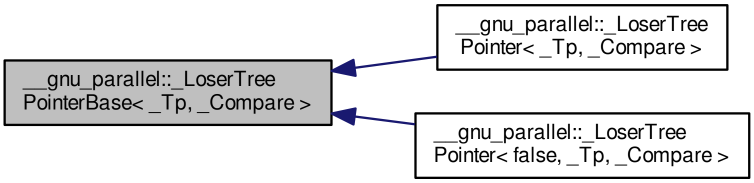
Definition at line 491 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.139 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >:



Classes

- struct _Loser

    Internal representation of _LoserTree __elements.

Public Member Functions

- **_LoserTreePointerBase** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.139.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >

Base class of _Loser Tree implementation using pointers.

Definition at line 357 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.140 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference

Public Attributes

- const _Tp ∗ **_M_keyp**
- int **_M_source**
- bool **_M_sup**

4.140.1 Detailed Description

template<typename _Tp, typename _Compare>struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser

Internal representation of _LoserTree __elements.

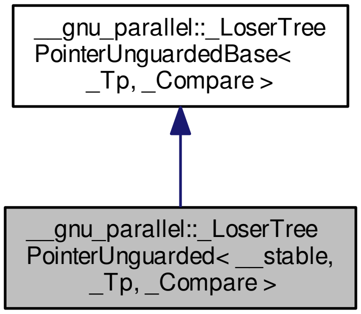
Definition at line 361 of file losertree.h.

The documentation for this struct was generated from the following file:

- losertree.h

4.141 __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.141.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >

Stable unguarded _LoserTree variant storing pointers.

Unstable variant is implemented below using partial specialization.

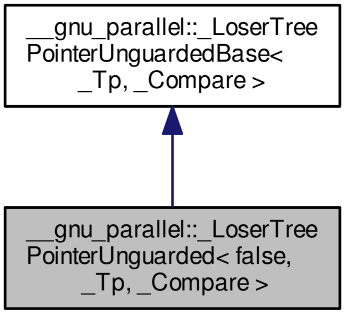
Definition at line 891 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.142 __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()

- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.142.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >

Unstable unguarded _LoserTree variant storing pointers.

Stable variant is above.

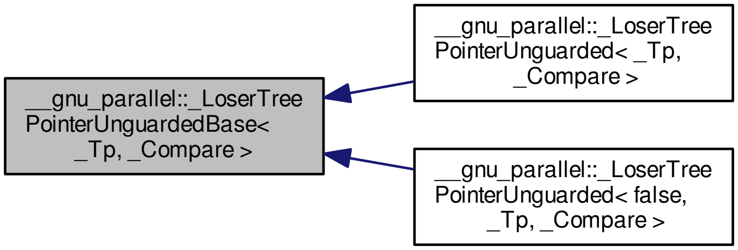
Definition at line 977 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.143 __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguardedBase** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.143.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 828 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.144 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference

Static Public Attributes

- static const bool _M_use_pointer

4.144.1 Detailed Description

template<typename _Tp>struct __gnu_parallel::_LoserTreeTraits< _Tp >

Traits for determining whether the loser tree should use pointers or copies.

The field "_M_use_pointer" is used to determine whether to use pointers in he loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };

template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };

Parameters

_Tp	type to give the loser tree traits for.

Definition at line 731 of file multiway_merge.h.

4.144.2 Member Data Documentation

4.144.2.1 template<typename _Tp > const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer [static]

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

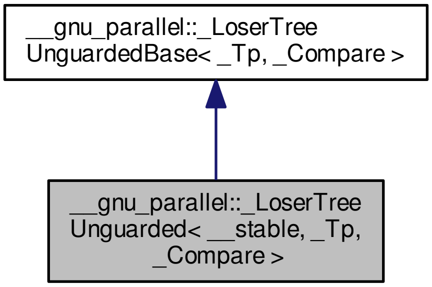
Definition at line 739 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.145 __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.145.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >

Stable implementation of unguarded _LoserTree.

Unstable variant is selected below with partial specialization.

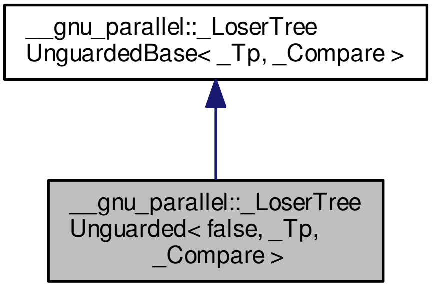
Definition at line 646 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.146 __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.146.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >

Non-Stable implementation of unguarded _LoserTree.

Stable implementation is above.

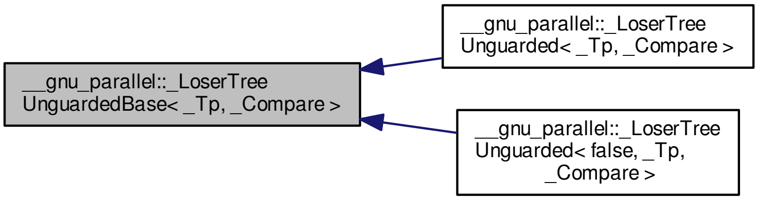
Definition at line 734 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.147 __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference

Inheritance diagram for __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguardedBase** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser ∗ **_M_losers**
- unsigned int **_M_offset**

4.147.1 Detailed Description

template<typename _Tp, typename _Compare>class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >

Base class for unguarded _LoserTree implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused __sequence heads are marked with a sentinel which is > all elements that are to be merged.

This is a very fast variant.

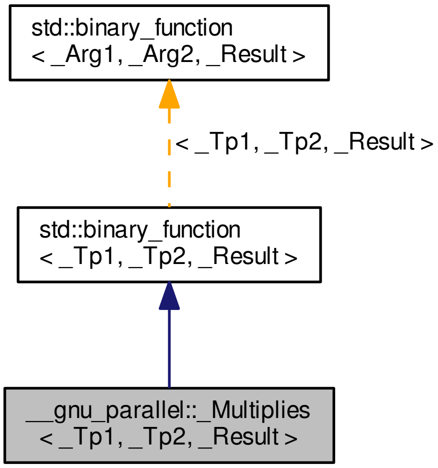
Definition at line 574 of file losertree.h.

The documentation for this class was generated from the following file:

- losertree.h

4.148 __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference

Inheritance diagram for __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >:



Public Types

- typedef _Tp1 first_argument_type
- typedef _Result result_type
- typedef _Tp2 second_argument_type

Public Member Functions

- _Result **operator()** (const _Tp1 &__x, const _Tp2 &__y) const

4.148.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))>struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >

Similar to std::multiplies, but allows two different types.

Definition at line 288 of file parallel/base.h.

4.148.2 Member Typedef Documentation

4.148.2.1 typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.148.2.2 typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.148.2.3 typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- parallel/base.h

4.149 __gnu_parallel::_Nothing Struct Reference

Public Member Functions

- template<typename _It >
  void operator() (_It __i)

4.149.1 Detailed Description

Functor doing nothing.

For some __reduction tasks (this is not a function object, but is passed as __selector __dummy parameter.

Definition at line 288 of file for_each_selectors.h.

4.149.2 Member Function Documentation

4.149.2.1 template<typename _It > void __gnu_parallel::_Nothing::operator() (_It *__i*) [inline]

Functor execution.

Parameters

__i	iterator referencing object.

Definition at line 294 of file for_each_selectors.h.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

4.150 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Attributes

- _DifferenceType _M_begin
- _DifferenceType _M_end

4.150.1 Detailed Description

template<typename _DifferenceTp>struct __gnu_parallel::_Piece< _DifferenceTp >

Subsequence description.

Definition at line 46 of file multiway_mergesort.h.

4.150.2 Member Data Documentation

4.150.2.1 template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin

Begin of subsequence.

Definition at line 51 of file multiway_mergesort.h.

4.150.2.2 template<typename _DifferenceTp > _DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end

End of subsequence.

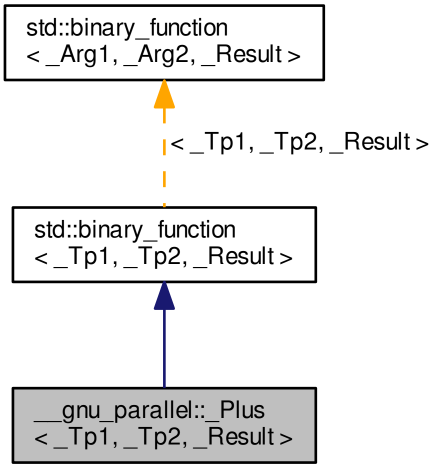
Definition at line 54 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- multiway_mergesort.h

4.151 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference

Inheritance diagram for __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >:



Public Types

- typedef _Tp1 first_argument_type
- typedef _Result result_type
- typedef _Tp2 second_argument_type

Public Member Functions

- _Result **operator()** (const _Tp1 &__x, const _Tp2 &__y) const

4.151.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))>struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >

Similar to std::plus, but allows two different types.

Definition at line 272 of file parallel/base.h.

4.151.2 Member Typedef Documentation

4.151.2.1 typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.151.2.2 typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.151.2.3 typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- parallel/base.h

4.152 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::iterator_traits < _RAIter > **_TraitsType**
- typedef _TraitsType::value_type **_ValueType**

Public Attributes

- _ThreadIndex _M_num_threads
- _DifferenceType ∗ _M_offsets
- std::vector< _Piece < _DifferenceType > > ∗ _M_pieces
- _ValueType ∗ _M_samples
- _RAIter _M_source
- _DifferenceType ∗ _M_starts
- _ValueType ∗∗ _M_temporary

4.152.1 Detailed Description

template<typename _RAIter>struct __gnu_parallel::_PMWMSSortingData< _RAIter >

Data accessed by all threads.

PMWMS = parallel multiway mergesort

Definition at line 61 of file multiway_mergesort.h.

4.152.2 Member Data Documentation

4.152.2.1 template<typename _RAIter> _ThreadIndex __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads

Number of threads involved.

Definition at line 68 of file multiway_mergesort.h.

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

4.152.2.2 template<typename _RAIter> _DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_offsets

Offsets to add to the found positions.

Definition at line 83 of file multiway_mergesort.h.

Referenced by __gnu_parallel::parallel_sort_mwms().

4.152.2.3 template<typename _RAIter> std::vector<_Piece<_DifferenceType> >* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces

Pieces of data to merge [thread][__sequence].

Definition at line 86 of file multiway_mergesort.h.

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

4.152.2.4 template<typename _RAIter> _ValueType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples

Samples.

Definition at line 80 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), and __gnu_parallel::parallel_sort_mwms().

4.152.2.5 template<typename _RAIter> _RAIter __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source

Input __begin.

Definition at line 71 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

4.152.2.6 template<typename _RAIter> _DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts

Start indices, per thread.

Definition at line 74 of file multiway_mergesort.h.

Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

4.152.2.7 template<typename _RAIter> _ValueType** __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary

Storage in which to sort.

Definition at line 77 of file multiway_mergesort.h.

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

The documentation for this struct was generated from the following file:

- multiway_mergesort.h

4.153 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference

Public Types

- typedef _DifferenceTp **_DifferenceType**
- typedef
  _PseudoSequenceIterator< _Tp,
  uint64_t > **iterator**

Public Member Functions

- _PseudoSequence (const _Tp &__val, _DifferenceType __count)
- iterator begin () const
- iterator end () const

4.153.1 Detailed Description

template<typename _Tp, typename _DifferenceTp>class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

_Tp	Sequence _M_value type.
_DifferenceTp	Sequence difference type.

Definition at line 359 of file parallel/base.h.

4.153.2 Constructor & Destructor Documentation

4.153.2.1 template<typename _Tp, typename _DifferenceTp> __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::_PseudoSequence (const _Tp & *__val,* _DifferenceType *__count*) [inline]

Constructor.

Parameters

__val	Element of the sequence.
__count	Number of (virtual) copies.

Definition at line 371 of file parallel/base.h.

4.153.3 Member Function Documentation

4.153.3.1 template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin ( ) const [inline]

Begin iterator.

Definition at line 376 of file parallel/base.h.

4.153.3.2 template<typename _Tp, typename _DifferenceTp> iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end ( ) const [inline]

End iterator.

Definition at line 381 of file parallel/base.h.

The documentation for this class was generated from the following file:

- parallel/base.h

4.154 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Member Functions

- **_PseudoSequenceIterator** (const _Tp &__val, _DifferenceType __pos)
- bool **operator!=** (const _PseudoSequenceIterator &__i2)
- const _Tp & **operator*** () const
- _PseudoSequenceIterator & **operator++** ()
- _PseudoSequenceIterator **operator++** (int)
- _DifferenceType **operator-** (const _PseudoSequenceIterator &__i2)
- bool **operator==** (const _PseudoSequenceIterator &__i2)
- const _Tp & **operator[ ]** (_DifferenceType) const

4.154.1 Detailed Description

template<typename _Tp, typename _DifferenceTp>class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >

_Iterator associated with __gnu_parallel::_PseudoSequence. If features the usual random-access iterator functionality.

Parameters

_Tp	Sequence _M_value type.
_DifferenceTp	Sequence difference type.

Definition at line 306 of file parallel/base.h.

The documentation for this class was generated from the following file:

- parallel/base.h

4.155 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference

Public Types

- typedef
  _TraitsType::difference_type **_DifferenceType**
- typedef std::pair< _RAIter,
  _RAIter > _Piece

- typedef std::iterator_traits
  < _RAIter > **_TraitsType**

Public Member Functions

- _QSBThreadLocal (int __queue_size)

Public Attributes

- volatile _DifferenceType ∗ _M_elements_leftover
- _Piece _M_global
- _Piece _M_initial
- _RestrictedBoundedConcurrentQueue
  < _Piece > _M_leftover_parts
- _ThreadIndex _M_num_threads

4.155.1 Detailed Description

template<typename _RAIter>struct __gnu_parallel::_QSBThreadLocal< _RAIter >

Information local to one thread in the parallel quicksort run.

Definition at line 62 of file balanced_quicksort.h.

4.155.2 Member Typedef Documentation

4.155.2.1 template<typename _RAIter> typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal< _RAIter >::_Piece

Continuous part of the sequence, described by an iterator pair.

Definition at line 69 of file balanced_quicksort.h.

4.155.3 Constructor & Destructor Documentation

4.155.3.1 template<typename _RAIter> __gnu_parallel::_QSBThreadLocal< _RAIter >::_QSBThreadLocal (int *__queue_size*) [inline]

Constructor.

Parameters

__queue_size	size of the work-stealing queue.

Definition at line 88 of file balanced_quicksort.h.

4.155.4 Member Data Documentation

4.155.4.1 template<typename _RAIter> volatile _DifferenceType∗ __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover

Pointer to a counter of elements left over to sort.

Definition at line 81 of file balanced_quicksort.h.

Referenced by __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__qsb_conquer(), and __gnu_parallel::__qsb_local_sort_with_helping().

4.155.4.2 template<typename _RAIter> _Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_global

The complete sequence to sort.

Definition at line 84 of file balanced_quicksort.h.

4.155.4.3 template<typename _RAIter> _Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial

Initial piece to work on.

Definition at line 72 of file balanced_quicksort.h.

Referenced by __gnu_parallel::__qsb_conquer(), and __gnu_parallel::__qsb_local_sort_with_helping().

4.155.4.4 template<typename _RAIter> _RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_leftover_parts

Work-stealing queue.

Definition at line 75 of file balanced_quicksort.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

4.155.4.5 template<typename _RAIter> _ThreadIndex __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads

Number of threads involved in this algorithm.

Definition at line 78 of file balanced_quicksort.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

The documentation for this struct was generated from the following file:

- balanced_quicksort.h

4.156 __gnu_parallel::_RandomNumber Class Reference

Public Member Functions

- _RandomNumber ()
- _RandomNumber (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long __genrand_bits (int __bits)
- uint32_t operator() ()
- uint32_t operator() (uint64_t local_supremum)

4.156.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file random_number.h.

4.156.2 Constructor & Destructor Documentation

4.156.2.1 __gnu_parallel::_RandomNumber::_RandomNumber ( ) [inline]

Default constructor. Seed with 0.

Definition at line 74 of file random_number.h.

4.156.2.2 __gnu_parallel::_RandomNumber::_RandomNumber (uint32_t *__seed*, uint64_t *_M_supremum* = 0x100000000ULL) [inline]

Constructor.

Parameters

__seed	Random __seed.
_M_supremum	Generate integer random numbers in the interval [0,_M_supremum).

Definition at line 85 of file random_number.h.

4.156.3 Member Function Documentation

4.156.3.1 unsigned long __gnu_parallel::_RandomNumber::__genrand_bits (int *__bits*) [inline]

Generate a number of random bits, run-time parameter.

Parameters

__bits	Number of bits to generate.

Definition at line 109 of file random_number.h.

4.156.3.2 uint32_t __gnu_parallel::_RandomNumber::operator() ( ) [inline]

Generate unsigned random 32-bit integer.

Definition at line 94 of file random_number.h.

4.156.3.3 uint32_t __gnu_parallel::_RandomNumber::operator() (uint64_t *local_supremum*) [inline]

Generate unsigned random 32-bit integer in the interval [0,local_supremum).

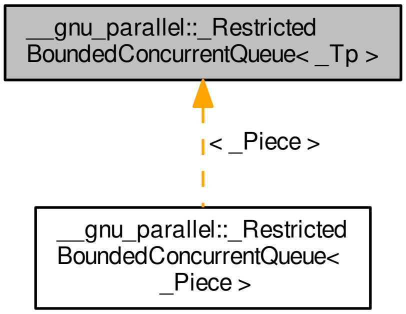
Definition at line 100 of file random_number.h.

The documentation for this class was generated from the following file:

- random_number.h

4.157 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference

Inheritance diagram for __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >:



Public Member Functions

- _RestrictedBoundedConcurrentQueue (_SequenceIndex __max_size)
- ∼_RestrictedBoundedConcurrentQueue ()
- bool pop_back (_Tp &__t)
- bool pop_front (_Tp &__t)
- void push_front (const _Tp &__t)

4.157.1 Detailed Description

template<typename _Tp>class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >

Double-ended queue of bounded size, allowing lock-free atomic access. push_front() and pop_front() must not be called concurrently to each other, while pop_back() can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

_Tp	Contained element type.

Definition at line 52 of file queue.h.

4.157.2 Constructor & Destructor Documentation

4.157.2.1 template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue (_SequenceIndex *__max_size*) `[inline]`

Constructor. Not to be called concurrent, of course.

Parameters

__max_size	Maximal number of elements to be contained.

Definition at line 68 of file queue.h.

4.157.2.2 template<typename _Tp> __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::~_RestrictedBoundedConcurrentQueue ( ) [inline]

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

4.157.3 Member Function Documentation

4.157.3.1 template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (_Tp & __t) [inline]

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().

Definition at line 127 of file queue.h.

4.157.3.2 template<typename _Tp> bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (_Tp & __t) [inline]

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().

Definition at line 100 of file queue.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

4.157.3.3 template<typename _Tp> void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (const _Tp & __t) [inline]

Pushes one element into the queue at the front end. Must not be called concurrently with pop_front().

Definition at line 83 of file queue.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

The documentation for this class was generated from the following file:

- queue.h

4.158 __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

4.158.1 Detailed Description

template<bool __stable, class _RAIter, class _StrictWeakOrdering>struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >

Stable sorting functor.

Used to reduce code instanciation in multiway_merge_sampling_splitting.

Definition at line 1007 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.159 __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

4.159.1 Detailed Description

template<class _RAIter, class _StrictWeakOrdering>struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >

Non-__stable sorting functor.

Used to reduce code instantiation in multiway_merge_sampling_splitting.

Definition at line 1020 of file multiway_merge.h.

The documentation for this struct was generated from the following file:

- multiway_merge.h

4.160 __gnu_parallel::_Settings Struct Reference

Static Public Member Functions

- static const _Settings & get () throw ()
- static void set (_Settings &) throw ()

Public Attributes

- _SequenceIndex accumulate_minimal_n
- unsigned int adjacent_difference_minimal_n
- _AlgorithmStrategy **algorithm_strategy**
- unsigned int cache_line_size
- _SequenceIndex count_minimal_n
- _SequenceIndex fill_minimal_n
- _FindAlgorithm **find_algorithm**
- double find_increasing_factor
- _SequenceIndex find_initial_block_size
- _SequenceIndex find_maximum_block_size
- float find_scale_factor
- _SequenceIndex find_sequential_search_size
- _SequenceIndex for_each_minimal_n
- _SequenceIndex generate_minimal_n
- unsigned long long L1_cache_size

- unsigned long long L2_cache_size
- _SequenceIndex max_element_minimal_n
- _SequenceIndex merge_minimal_n
- unsigned int merge_oversampling
- _SplittingAlgorithm **merge_splitting**
- _SequenceIndex min_element_minimal_n
- _MultiwayMergeAlgorithm **multiway_merge_algorithm**
- int multiway_merge_minimal_k
- _SequenceIndex multiway_merge_minimal_n
- unsigned int multiway_merge_oversampling
- _SplittingAlgorithm **multiway_merge_splitting**
- _SequenceIndex nth_element_minimal_n
- _SequenceIndex partial_sort_minimal_n
- _PartialSumAlgorithm **partial_sum_algorithm**
- float partial_sum_dilation
- unsigned int partial_sum_minimal_n
- double partition_chunk_share
- _SequenceIndex partition_chunk_size
- _SequenceIndex partition_minimal_n
- _SequenceIndex qsb_steals
- unsigned int random_shuffle_minimal_n
- _SequenceIndex replace_minimal_n
- _SequenceIndex search_minimal_n
- _SequenceIndex set_difference_minimal_n
- _SequenceIndex set_intersection_minimal_n
- _SequenceIndex set_symmetric_difference_minimal_n
- _SequenceIndex set_union_minimal_n
- _SortAlgorithm **sort_algorithm**
- _SequenceIndex sort_minimal_n
- unsigned int sort_mwms_oversampling
- unsigned int sort_qs_num_samples_preset
- _SequenceIndex sort_qsb_base_case_maximal_n
- _SplittingAlgorithm **sort_splitting**
- unsigned int TLB_size
- _SequenceIndex transform_minimal_n
- _SequenceIndex unique_copy_minimal_n
- _SequenceIndex **workstealing_chunk_size**

4.160.1 Detailed Description

class _Settings Run-time settings for the parallel mode including all tunable parameters.

Definition at line 123 of file settings.h.

4.160.2 Member Function Documentation

4.160.2.1 static const _Settings& __gnu_parallel::_Settings::get ( ) throw () [static]

Get the global settings.

Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_parallel::__parallel_nth_element(), __gnu_parallel::__parallel_partial_sum(), __gnu_parallel::__parallel_partial_sum_linear(), __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort_qs_conquer(), __gnu_parallel::__qsb_local_sort_with_helping(), __gnu_parallel::__sequential_random_shuffle(), __gnu_parallel::multiway_merge_sampling_splitting(), __gnu_parallel::parallel_multiway_merge(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

4.160.2.2 static void __gnu_parallel::_Settings::set (_Settings &) throw () [static]

Set the global settings.

4.160.3 Member Data Documentation

4.160.3.1 _SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n

Minimal input size for accumulate.

Definition at line 139 of file settings.h.

4.160.3.2 unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n

Minimal input size for adjacent_difference.

Definition at line 142 of file settings.h.

4.160.3.3 unsigned int __gnu_parallel::_Settings::cache_line_size

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Definition at line 265 of file settings.h.

Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

4.160.3.4 _SequenceIndex __gnu_parallel::_Settings::count_minimal_n

Minimal input size for count and count_if.

Definition at line 145 of file settings.h.

4.160.3.5 _SequenceIndex __gnu_parallel::_Settings::fill_minimal_n

Minimal input size for fill.

Definition at line 148 of file settings.h.

4.160.3.6 double __gnu_parallel::_Settings::find_increasing_factor

Block size increase factor for find.

Definition at line 151 of file settings.h.

4.160.3.7 _SequenceIndex __gnu_parallel::_Settings::find_initial_block_size

Initial block size for find.

Definition at line 154 of file settings.h.

Referenced by __gnu_parallel::__find_template().

4.160.3.8 _SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size

Maximal block size for find.

Definition at line 157 of file settings.h.

4.160.3.9 float __gnu_parallel::_Settings::find_scale_factor

Block size scale-down factor with respect to current position.

Definition at line 276 of file settings.h.

Referenced by __gnu_parallel::__find_template().

4.160.3.10 _SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size

Start with looking for this many elements sequentially, for find.

Definition at line 160 of file settings.h.

Referenced by __gnu_parallel::__find_template().

4.160.3.11 _SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n

Minimal input size for for_each.

Definition at line 163 of file settings.h.

4.160.3.12 _SequenceIndex __gnu_parallel::_Settings::generate_minimal_n

Minimal input size for generate.

Definition at line 166 of file settings.h.

4.160.3.13 unsigned long long __gnu_parallel::_Settings::L1_cache_size

size of the L1 cache in bytes (underestimation).

Definition at line 254 of file settings.h.

4.160.3.14 unsigned long long __gnu_parallel::_Settings::L2_cache_size

size of the L2 cache in bytes (underestimation).

Definition at line 257 of file settings.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__sequential_random_shuffle().

4.160.3.15 _SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n

Minimal input size for max_element.

Definition at line 169 of file settings.h.

4.160.3.16 _SequenceIndex __gnu_parallel::_Settings::merge_minimal_n

Minimal input size for merge.

Definition at line 172 of file settings.h.

4.160.3.17 unsigned int __gnu_parallel::_Settings::merge_oversampling

Oversampling factor for merge.

Definition at line 175 of file settings.h.

Referenced by __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiway_merge().

4.160.3.18 _SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n

Minimal input size for min_element.

Definition at line 178 of file settings.h.

4.160.3.19 int __gnu_parallel::_Settings::multiway_merge_minimal_k

Oversampling factor for multiway_merge.

Definition at line 184 of file settings.h.

4.160.3.20 _SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n

Minimal input size for multiway_merge.

Definition at line 181 of file settings.h.

4.160.3.21 unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling

Oversampling factor for multiway_merge.

Definition at line 187 of file settings.h.

4.160.3.22 _SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n

Minimal input size for nth_element.

Definition at line 190 of file settings.h.

Referenced by __gnu_parallel::__parallel_nth_element().

4.160.3.23 _SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n

Minimal input size for partial_sort.

Definition at line 203 of file settings.h.

4.160.3.24 float __gnu_parallel::_Settings::partial_sum_dilation

Ratio for partial_sum. Assume "sum and write result" to be this factor slower than just "sum".

Definition at line 207 of file settings.h.

Referenced by __gnu_parallel::__parallel_partial_sum_linear().

4.160.3.25 unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n

Minimal input size for partial_sum.

Definition at line 210 of file settings.h.

4.160.3.26 double __gnu_parallel::_Settings::partition_chunk_share

Chunk size for partition, relative to input size. If > 0.0, this value overrides partition_chunk_size.

Definition at line 197 of file settings.h.

Referenced by __gnu_parallel::__parallel_partition().

4.160.3.27 _SequenceIndex __gnu_parallel::_Settings::partition_chunk_size

Chunk size for partition.

Definition at line 193 of file settings.h.

Referenced by __gnu_parallel::__parallel_partition().

4.160.3.28 _SequenceIndex __gnu_parallel::_Settings::partition_minimal_n

Minimal input size for partition.

Definition at line 200 of file settings.h.

Referenced by __gnu_parallel::__parallel_nth_element().

4.160.3.29 _SequenceIndex __gnu_parallel::_Settings::qsb_steals

The number of stolen ranges in load-balanced quicksort.

Definition at line 270 of file settings.h.

4.160.3.30 unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n

Minimal input size for random_shuffle.

Definition at line 213 of file settings.h.

4.160.3.31 _SequenceIndex __gnu_parallel::_Settings::replace_minimal_n

Minimal input size for replace and replace_if.

Definition at line 216 of file settings.h.

4.160.3.32 _SequenceIndex __gnu_parallel::_Settings::search_minimal_n

Minimal input size for search and search_n.

Definition at line 273 of file settings.h.

4.160.3.33 _SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n

Minimal input size for set_difference.

Definition at line 219 of file settings.h.

4.160.3.34 _SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n

Minimal input size for set_intersection.

Definition at line 222 of file settings.h.

4.160.3.35 _SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n

Minimal input size for set_symmetric_difference.

Definition at line 225 of file settings.h.

4.160.3.36 _SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n

Minimal input size for set_union.

Definition at line 228 of file settings.h.

4.160.3.37 _SequenceIndex __gnu_parallel::_Settings::sort_minimal_n

Minimal input size for parallel sorting.

Definition at line 231 of file settings.h.

4.160.3.38 unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling

Oversampling factor for parallel std::sort (MWMS).

Definition at line 234 of file settings.h.

Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

4.160.3.39 unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset

Such many samples to take to find a good pivot (quicksort).

Definition at line 237 of file settings.h.

4.160.3.40 _SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n

Maximal subsequence __length to switch to unbalanced __base case. Applies to std::sort with dynamically load-balanced quicksort.

Definition at line 241 of file settings.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

4.160.3.41 unsigned int __gnu_parallel::_Settings::TLB_size

size of the Translation Lookaside Buffer (underestimation).

Definition at line 260 of file settings.h.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__sequential_random_shuffle().

4.160.3.42 _SequenceIndex __gnu_parallel::_Settings::transform_minimal_n

Minimal input size for parallel std::transform.

Definition at line 244 of file settings.h.

4.160.3.43 _SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n

Minimal input size for unique_copy.

Definition at line 247 of file settings.h.

The documentation for this struct was generated from the following file:

- settings.h

4.161 __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

4.161.1 Detailed Description

template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >

Split consistently.

Definition at line 122 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- multiway_mergesort.h

4.162 __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

Public Member Functions

- void **operator()** (const _ThreadIndex __iam, _PMWMSSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

4.162.1 Detailed Description

template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >

Split by sampling.

Definition at line 187 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- multiway_mergesort.h

4.163 __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

Public Member Functions

- void **operator()** (const _ThreadIndex __iam, _PMWMSSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

4.163.1 Detailed Description

template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

Split by exact splitting.

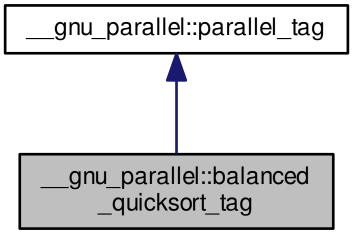
Definition at line 128 of file multiway_mergesort.h.

The documentation for this struct was generated from the following file:

- multiway_mergesort.h

4.164 __gnu_parallel::balanced_quicksort_tag Struct Reference

Inheritance diagram for __gnu_parallel::balanced_quicksort_tag:



Public Member Functions

- **balanced_quicksort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.164.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

4.164.2 Member Function Documentation

4.164.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.164.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

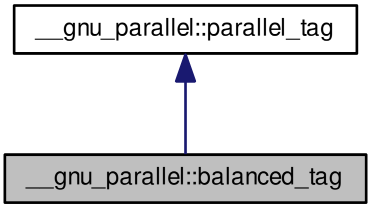
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.165 __gnu_parallel::balanced_tag Struct Reference

Inheritance diagram for __gnu_parallel::balanced_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.165.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

4.165.2 Member Function Documentation

4.165.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.165.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

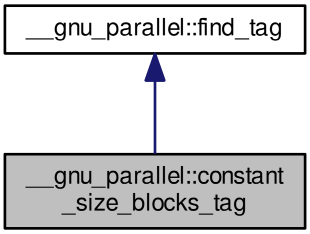
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.166 __gnu_parallel::constant_size_blocks_tag Struct Reference

Inheritance diagram for __gnu_parallel::constant_size_blocks_tag:



4.166.1 Detailed Description

Selects the constant block size variant for std::find().

See Also

_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS

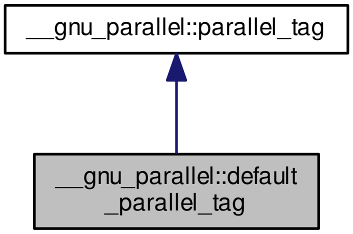
Definition at line 178 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.167 __gnu_parallel::default_parallel_tag Struct Reference

Inheritance diagram for __gnu_parallel::default_parallel_tag:



Public Member Functions

- **default_parallel_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.167.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file tags.h.

4.167.2 Member Function Documentation

4.167.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) `[inline]`, `[inherited]`

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.167.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex __num_threads) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

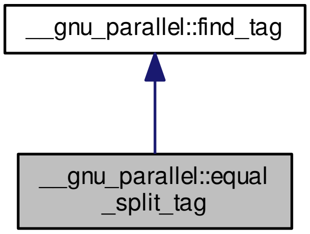
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.168 __gnu_parallel::equal_split_tag Struct Reference

Inheritance diagram for __gnu_parallel::equal_split_tag:



4.168.1 Detailed Description

Selects the equal splitting variant for std::find().

See Also

_GLIBCXX_FIND_EQUAL_SPLIT

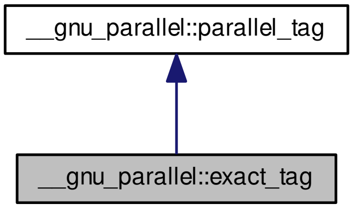
Definition at line 182 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.169 __gnu_parallel::exact_tag Struct Reference

Inheritance diagram for __gnu_parallel::exact_tag:



Public Member Functions

- **exact_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.169.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file tags.h.

4.169.2 Member Function Documentation

4.169.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.169.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

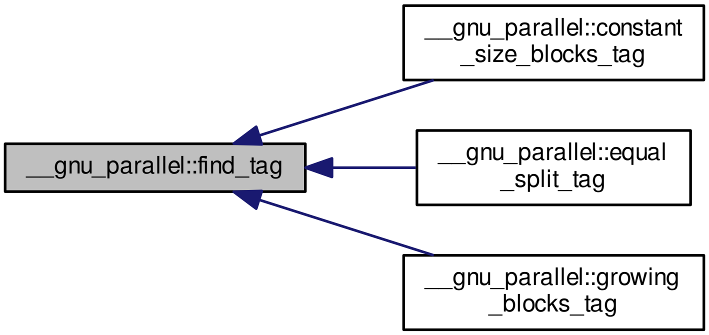
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.170 __gnu_parallel::find_tag Struct Reference

Inheritance diagram for __gnu_parallel::find_tag:



4.170.1 Detailed Description

Base class for for std::find() variants.

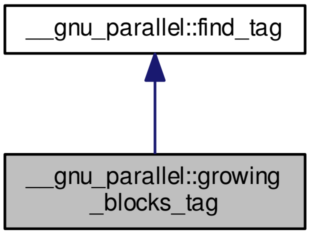
Definition at line 104 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.171 __gnu_parallel::growing_blocks_tag Struct Reference

Inheritance diagram for __gnu_parallel::growing_blocks_tag:



4.171.1 Detailed Description

Selects the growing block size variant for std::find().

See Also

_GLIBCXX_FIND_GROWING_BLOCKS

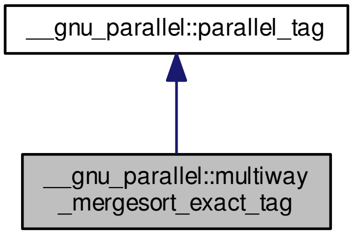
Definition at line 174 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.172 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference

Inheritance diagram for __gnu_parallel::multiway_mergesort_exact_tag:



Public Member Functions

- **multiway_mergesort_exact_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.172.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

4.172.2 Member Function Documentation

4.172.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.172.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

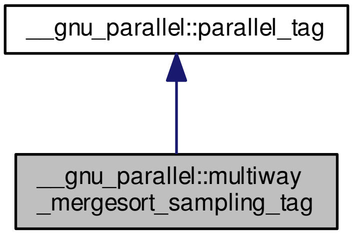
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.173 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference

Inheritance diagram for __gnu_parallel::multiway_mergesort_sampling_tag:



Public Member Functions

- **multiway_mergesort_sampling_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.173.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

4.173.2 Member Function Documentation

4.173.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.173.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

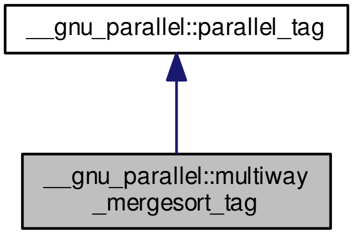
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.174 __gnu_parallel::multiway_mergesort_tag Struct Reference

Inheritance diagram for __gnu_parallel::multiway_mergesort_tag:



Public Member Functions

- **multiway_mergesort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.174.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

4.174.2 Member Function Documentation

4.174.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.174.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

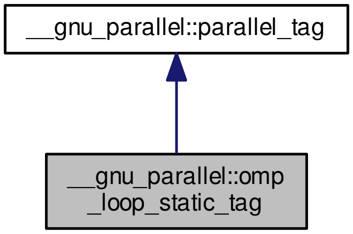
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.175 __gnu_parallel::omp_loop_static_tag Struct Reference

Inheritance diagram for __gnu_parallel::omp_loop_static_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.175.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

4.175.2 Member Function Documentation

4.175.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.175.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

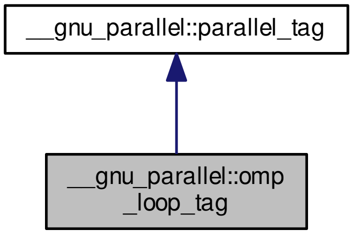
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.176 __gnu_parallel::omp_loop_tag Struct Reference

Inheritance diagram for __gnu_parallel::omp_loop_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.176.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

4.176.2 Member Function Documentation

4.176.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.176.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

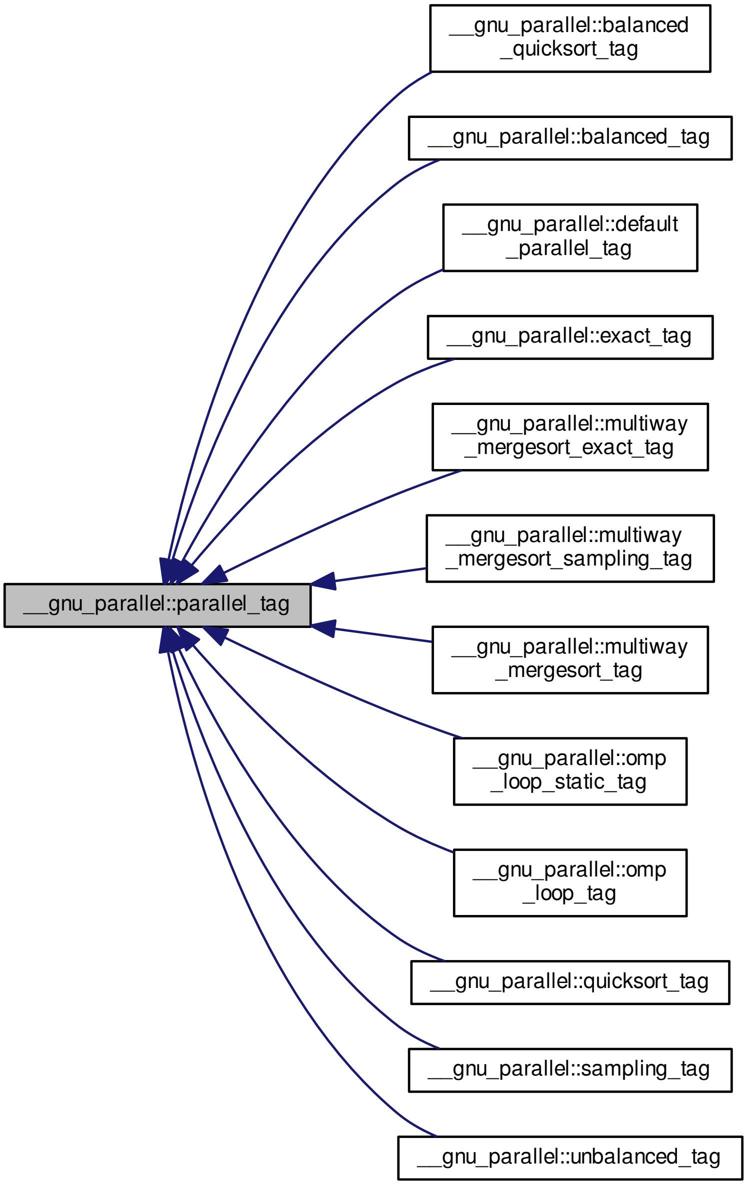
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.177 __gnu_parallel::parallel_tag Struct Reference

Inheritance diagram for __gnu_parallel::parallel_tag:



Public Member Functions

- parallel_tag ()
- parallel_tag (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()

- void set_num_threads (_ThreadIndex __num_threads)

4.177.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

4.177.2 Constructor & Destructor Documentation

4.177.2.1 __gnu_parallel::parallel_tag::parallel_tag ( ) [inline]

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

4.177.2.2 __gnu_parallel::parallel_tag::parallel_tag (_ThreadIndex *__num_threads*) [inline]

Default constructor. Recommend number of threads to use.

Parameters

__num_threads	Desired number of threads.

Definition at line 58 of file tags.h.

4.177.3 Member Function Documentation

4.177.3.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.177.3.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

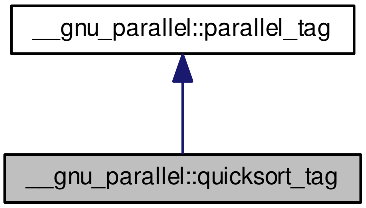
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.178 __gnu_parallel::quicksort_tag Struct Reference

Inheritance diagram for __gnu_parallel::quicksort_tag:



Public Member Functions

- **quicksort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.178.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

4.178.2 Member Function Documentation

4.178.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.178.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

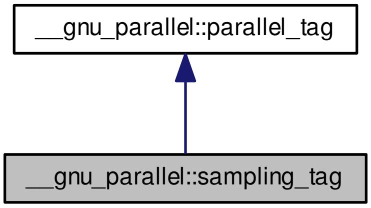
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.179 __gnu_parallel::sampling_tag Struct Reference

Inheritance diagram for __gnu_parallel::sampling_tag:



Public Member Functions

- **sampling_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.179.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

4.179.2 Member Function Documentation

4.179.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.179.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.180 __gnu_parallel::sequential_tag Struct Reference

4.180.1 Detailed Description

Forces sequential execution at compile time.

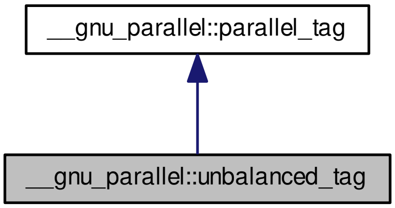
Definition at line 42 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.181 __gnu_parallel::unbalanced_tag Struct Reference

Inheritance diagram for __gnu_parallel::unbalanced_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

4.181.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

4.181.2 Member Function Documentation

4.181.2.1 _ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ( ) [inline], [inherited]

Find out desired number of threads.

Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by __gnu_parallel::__parallel_sort().

4.181.2.2 void __gnu_parallel::parallel_tag::set_num_threads (_ThreadIndex *__num_threads*) [inline], [inherited]

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

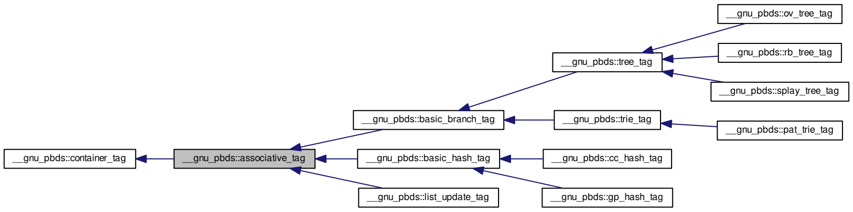
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- tags.h

4.182 __gnu_pbds::associative_tag Struct Reference

Inheritance diagram for __gnu_pbds::associative_tag:



4.182.1 Detailed Description

Basic associative-container.

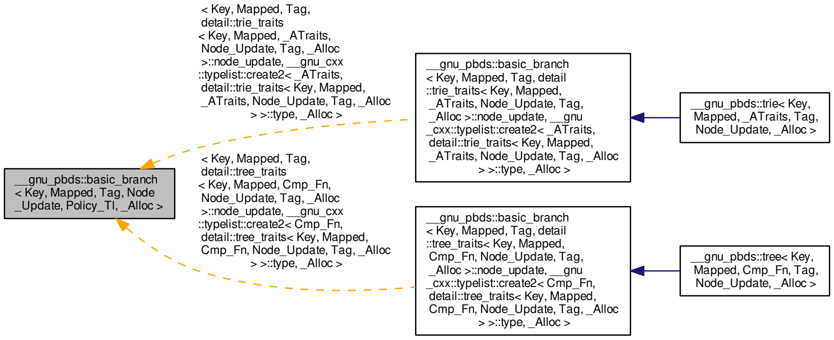
Definition at line 135 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.183 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >:



Public Types

- typedef Node_Update **node_update**

Protected Member Functions

- **basic_branch** (const basic_branch &other)
- template< typename T0 >
  basic_branch (T0 t0)
- template< typename T0 , typename T1 >
  basic_branch (T0 t0, T1 t1)
- template< typename T0 , typename T1 , typename T2 >
  basic_branch (T0 t0, T1 t1, T2 t2)
- template< typename T0 , typename T1 , typename T2 , typename T3 >
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

4.183.1 Detailed Description

template< typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc> class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >

A branched, tree-like (tree, trie) container abstraction.

Template Parameters

Key	Key type.
Mapped	Map type.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates nodes, restores invariants.
Policy_TL	Policy typelist.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: tree_tag, trie_tag, and their descendants.

Base choices are: detail::ov_tree_map, detail::rb_tree_map, detail::splay_tree_map, and detail::pat_trie_map.

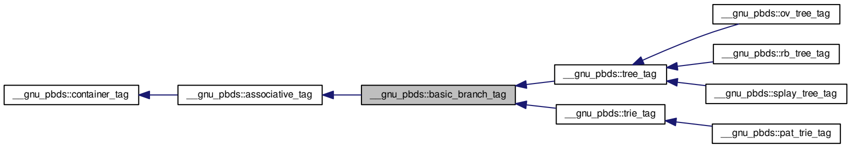
Definition at line 555 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.184 __gnu_pbds::basic_branch_tag Struct Reference

Inheritance diagram for __gnu_pbds::basic_branch_tag:



4.184.1 Detailed Description

Basic branch structure.

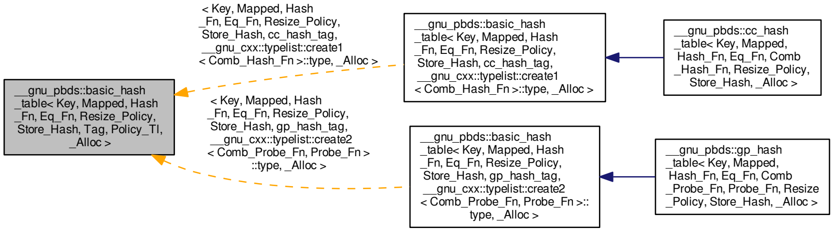
Definition at line 147 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.185 __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >:



Protected Member Functions

- **basic_hash_table** (const basic_hash_table &other)
- template< typename T0 >
  basic_hash_table (T0 t0)
- template< typename T0 , typename T1 >
  basic_hash_table (T0 t0, T1 t1)
- template< typename T0 , typename T1 , typename T2 >
  basic_hash_table (T0 t0, T1 t1, T2 t2)
- template< typename T0 , typename T1 , typename T2 , typename T3 >
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 >
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template< typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 , typename T7 , typename T8 >
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

4.185.1 Detailed Description

template< typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_Tl, typename _Alloc >class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >

A hashed container abstraction.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key.
Tag	Instantiating data structure type, see container_tag.
Policy_TL	Policy typelist.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: cc_hash_tag, gp_hash_tag, and descendants of basic_hash_tag.

Base choices are: detail::cc_ht_map, detail::gp_ht_map

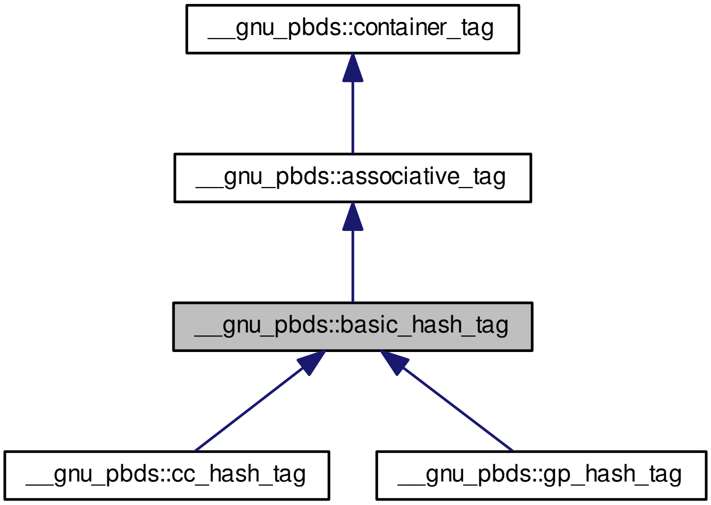
Definition at line 104 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.186 __gnu_pbds::basic_hash_tag Struct Reference

Inheritance diagram for __gnu_pbds::basic_hash_tag:



4.186.1 Detailed Description

Basic hash structure.

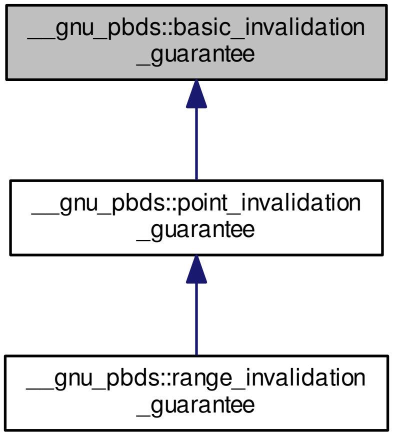
Definition at line 138 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.187 __gnu_pbds::basic_invalidation_guarantee Struct Reference

Inheritance diagram for __gnu_pbds::basic_invalidation_guarantee:



4.187.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

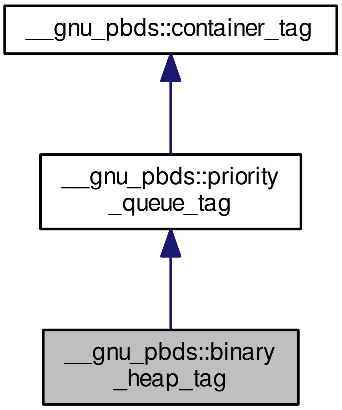
Definition at line 93 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.188 __gnu_pbds::binary_heap_tag Struct Reference

Inheritance diagram for __gnu_pbds::binary_heap_tag:



4.188.1 Detailed Description

Binary-heap (array-based).

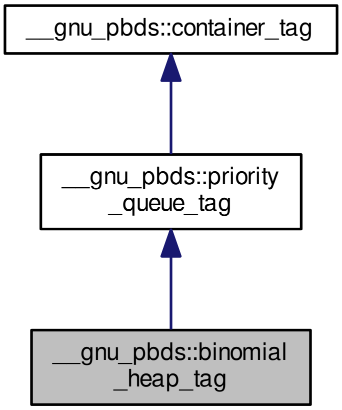
Definition at line 183 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.189 __gnu_pbds::binomial_heap_tag Struct Reference

Inheritance diagram for __gnu_pbds::binomial_heap_tag:



4.189.1 Detailed Description

Binomial-heap.

Definition at line 177 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.190 __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference

Public Types

- enum { external_load_access }
- typedef Size_Type **size_type**

Public Member Functions

- cc_hash_max_collision_check_resize_trigger (float load=0.5)
- float get_load () const
- void set_load (float load)
- void **swap** (cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > &other)

Protected Member Functions

- bool is_grow_needed (size_type size, size_type num_entries) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_entries)
- void notify_externally_resized (size_type new_size)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

4.190.1 Detailed Description

template<bool External_Load_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

Definition at line 293 of file hash_policy.hpp.

4.190.2 Member Enumeration Documentation

4.190.2.1 template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum

Enumerator:

external_load_access Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 298 of file hash_policy.hpp.

4.190.3 Constructor & Destructor Documentation

4.190.3.1 template<bool External_Load_Access, typename Size_Type > __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_hash_max_collision_check_resize_trigger (float *load* = `0.5`)

Default constructor, or constructor taking load, a __load factor which it will attempt to maintain.

Definition at line 44 of file hash_policy.hpp.

4.190.4 Member Function Documentation

4.190.4.1 template<bool External_Load_Access, typename Size_Type > float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::get_load ( ) const [inline]

Returns the current load.

Definition at line 190 of file hash_policy.hpp.

4.190.4.2 template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::is_grow_needed (size_type *size*, size_type *num_entries*) const [inline], [protected]

Queries whether a grow is needed. This method is called only if this object indicated is needed.

Definition at line 133 of file hash_policy.hpp.

4.190.4.3 template<bool External_Load_Access, typename Size_Type > bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::is_resize_needed ( ) const [inline], [protected]

Queries whether a resize is needed.

Definition at line 127 of file hash_policy.hpp.

4.190.4.4 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ( ) [protected]

Notifies the table was cleared.

Definition at line 121 of file hash_policy.hpp.

4.190.4.5 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

Definition at line 97 of file hash_policy.hpp.

4.190.4.6 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_end ( ) [inline], [protected]

Notifies a search ended.

Definition at line 103 of file hash_policy.hpp.

4.190.4.7 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erase_search_start ( ) [inline], [protected]

Notifies an erase search started.

Definition at line 91 of file hash_policy.hpp.

4.190.4.8 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_erased (size_type *num_entries*) [inline], [protected]

Notifies an element was erased.

Definition at line 115 of file hash_policy.hpp.

4.190.4.9 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_externally_resized (size_type *new_size*) [protected]

Notifies the table was resized externally.

Definition at line 172 of file hash_policy.hpp.

4.190.4.10 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

Definition at line 61 of file hash_policy.hpp.

4.190.4.11 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_end ( ) [inline], [protected]

Notifies a search ended.

Definition at line 67 of file hash_policy.hpp.

4.190.4.12 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_find_search_start ( ) [inline], [protected]

Notifies a find search started.

Definition at line 55 of file hash_policy.hpp.

4.190.4.13 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

Definition at line 79 of file hash_policy.hpp.

4.190.4.14 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_end ( ) [inline], [protected]

Notifies a search ended.

Definition at line 85 of file hash_policy.hpp.

4.190.4.15 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_insert_search_start ( ) [inline], [protected]

Notifies an insert search started.

Definition at line 73 of file hash_policy.hpp.

4.190.4.16 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted (size_type *num_entries*) [inline], [protected]

Notifies an element was inserted.

Definition at line 109 of file hash_policy.hpp.

4.190.4.17 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized (size_type *new_size*) [protected]

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 139 of file hash_policy.hpp.

4.190.4.18 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::set_load (float *load*)

Sets the load; does not resize the container.

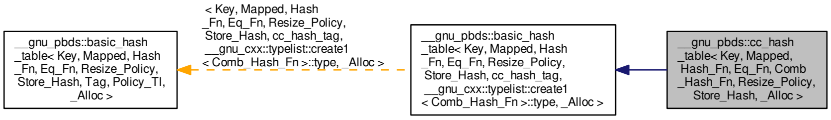
Definition at line 205 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- cc_hash_max_collision_check_resize_trigger_imp.hpp

4.191 __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >:



Public Types

- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef cc_hash_tag **container_category**

- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- cc_hash_table ()
- cc_hash_table (const hash_fn &h)
- cc_hash_table (const hash_fn &h, const eq_fn &e)
- cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- template<typename It >
  cc_hash_table (It first, It last)
- template<typename It >
  cc_hash_table (It first, It last, const hash_fn &h)
- template<typename It >
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It >
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- template<typename It >
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- **cc_hash_table** (const cc_hash_table &other)
- cc_hash_table & **operator=** (const cc_hash_table &other)
- void **swap** (cc_hash_table &other)

4.191.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >

A collision-chaining hash-based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Comb_Hash_Fn	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
_Alloc	Allocator type.

Base tag choices are: cc_hash_tag.

Base is basic_hash_table.

Definition at line 204 of file assoc_container.hpp.

4.191.2 Constructor & Destructor Documentation

4.191.2.1 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table ( ) [inline]

Default constructor.

Definition at line 217 of file assoc_container.hpp.

4.191.2.2 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & *h*) [inline]

Constructor taking some policy objects. r_hash_fn will be copied by the Hash_Fn object of the container object.

Definition at line 221 of file assoc_container.hpp.

4.191.2.3 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & *h*, const eq_fn & *e*) [inline]

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

Definition at line 228 of file assoc_container.hpp.

4.191.2.4 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & *h*, const eq_fn & *e*, const comb_hash_fn & *ch*) [inline]

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_hash_fn will be copied by the comb_hash_fn object of the container object.

Definition at line 236 of file assoc_container.hpp.

4.191.2.5 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (const hash_fn & *h*, const eq_fn & *e*, const comb_hash_fn & *ch*, const resize_policy & *rp*) `[inline]`

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_hash_fn will be copied by the comb_hash_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

Definition at line 245 of file assoc_container.hpp.

4.191.2.6 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (It *first*, It *last*) `[inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 253 of file assoc_container.hpp.

4.191.2.7 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (It *first*, It *last*, const hash_fn & *h*) `[inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 260 of file assoc_container.hpp.

4.191.2.8 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (It *first*, It *last*, const hash_fn & *h*, const eq_fn & *e*) `[inline]`

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

Definition at line 271 of file assoc_container.hpp.

4.191.2.9 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (It *first,* It *last,* const hash_fn & *h,* const eq_fn & *e,* const comb_hash_fn & *ch*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_hash_fn will be copied by the comb_hash_fn object of the container object.

Definition at line 283 of file assoc_container.hpp.

4.191.2.10 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (It *first,* It *last,* const hash_fn & *h,* const eq_fn & *e,* const comb_hash_fn & *ch,* const resize_policy & *rp*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_hash_fn will be copied by the comb_hash_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

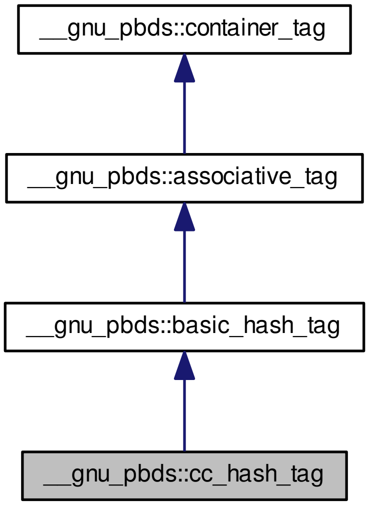
Definition at line 297 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.192 __gnu_pbds::cc_hash_tag Struct Reference

Inheritance diagram for __gnu_pbds::cc_hash_tag:



4.192.1 Detailed Description

Collision-chaining hash.

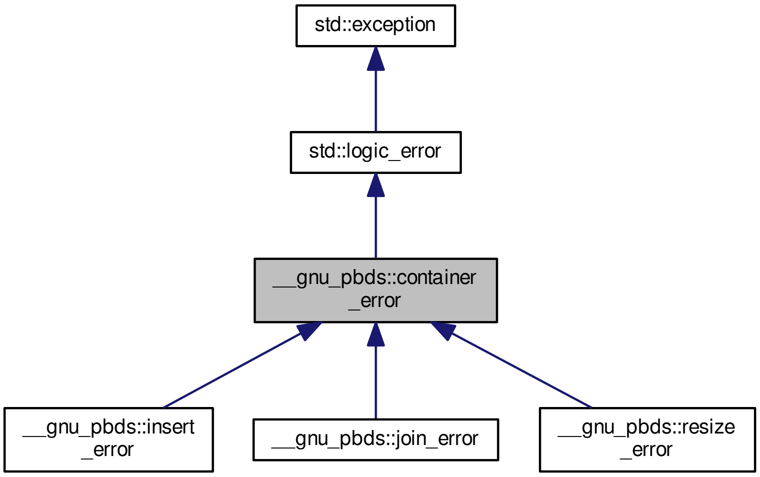
Definition at line 141 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.193 __gnu_pbds::container_error Struct Reference

Inheritance diagram for __gnu_pbds::container_error:



Public Member Functions

- virtual const char $*$ what () const noexcept

4.193.1 Detailed Description

Base class for exceptions.

Definition at line 57 of file exception.hpp.

4.193.2 Member Function Documentation

4.193.2.1 virtual const char$*$ std::logic_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

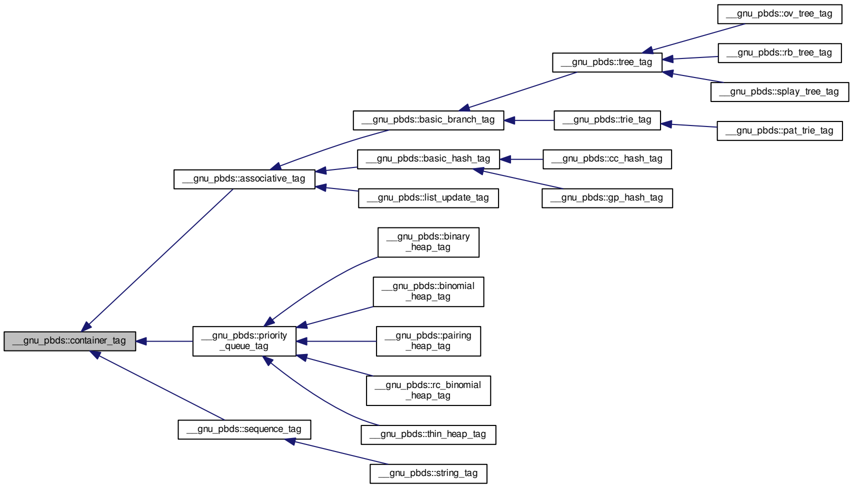
Reimplemented in std::future_error.

The documentation for this struct was generated from the following file:

- exception.hpp

4.194 __gnu_pbds::container_tag Struct Reference

Inheritance diagram for __gnu_pbds::container_tag:



4.194.1 Detailed Description

Base data structure tag.

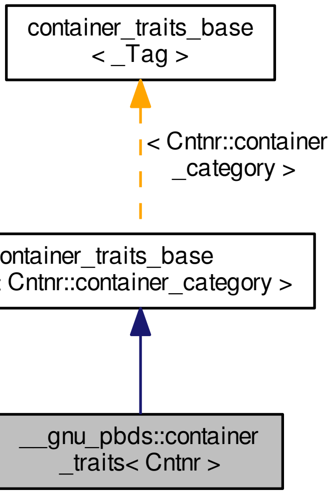
Definition at line 125 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.195 __gnu_pbds::container_traits< Cntnr > Struct Template Reference

Inheritance diagram for __gnu_pbds::container_traits< Cntnr >:



Public Types

- enum { order_preserving, erase_can_throw, split_join_can_throw, reverse_iteration }
- typedef container_traits_base
  < container_category > **base_type**
- typedef Cntnr::container_category **container_category**
- typedef Cntnr **container_type**
- typedef
  base_type::invalidation_guarantee **invalidation_guarantee**

4.195.1 Detailed Description

template<typename Cntnr>struct __gnu_pbds::container_traits< Cntnr >

Container traits.

Definition at line 418 of file tag_and_trait.hpp.

4.195.2 Member Enumeration Documentation

4.195.2.1 template<typename Cntnr > anonymous enum

Enumerator:

order_preserving True only if Cntnr objects guarantee storing keys by order.

erase_can_throw True only if erasing a key can throw.

split_join_can_throw True only if split or join operations can throw.

reverse_iteration True only reverse iterators are supported.

Definition at line 426 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.196 __gnu_pbds::container_traits_base< binary_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef binary_heap_tag **container_category**
- typedef
  basic_invalidation_guarantee **invalidation_guarantee**

4.196.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< binary_heap_tag >

Specialization, binary heap.

Definition at line 400 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.197 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef binomial_heap_tag **container_category**
- typedef
  point_invalidation_guarantee **invalidation_guarantee**

4.197.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< binomial_heap_tag >

Specialization, binomial heap.

Definition at line 368 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.198 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef cc_hash_tag **container_category**
- typedef
  point_invalidation_guarantee **invalidation_guarantee**

4.198.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< cc_hash_tag >

Specialization, cc hash.

Definition at line 224 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.199 __gnu_pbds::container_traits_base< gp_hash_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef gp_hash_tag **container_category**
- typedef
  basic_invalidation_guarantee **invalidation_guarantee**

4.199.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< gp_hash_tag >

Specialization, gp hash.

Definition at line 240 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.200 __gnu_pbds::container_traits_base< list_update_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef list_update_tag **container_category**
- typedef
  point_invalidation_guarantee **invalidation_guarantee**

4.200.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< list_update_tag >

Specialization, list update.

Definition at line 320 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.201 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef ov_tree_tag **container_category**
- typedef
  basic_invalidation_guarantee **invalidation_guarantee**

4.201.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< ov_tree_tag >

Specialization, ov tree.

Definition at line 288 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.202 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef pairing_heap_tag **container_category**
- typedef
  point_invalidation_guarantee **invalidation_guarantee**

4.202.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< pairing_heap_tag >

Specialization, pairing heap.

Definition at line 336 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.203 __gnu_pbds::container_traits_base< pat_trie_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef pat_trie_tag **container_category**
- typedef
  range_invalidation_guarantee **invalidation_guarantee**

4.203.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< pat_trie_tag >

Specialization, pat trie.

Definition at line 304 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.204 __gnu_pbds::container_traits_base< rb_tree_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef rb_tree_tag **container_category**
- typedef
  range_invalidation_guarantee **invalidation_guarantee**

4.204.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< rb_tree_tag >

Specialization, rb tree.

Definition at line 256 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.205 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef rc_binomial_heap_tag **container_category**
- typedef
  point_invalidation_guarantee **invalidation_guarantee**

4.205.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >

Specialization, rc binomial heap.

Definition at line 384 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.206 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef splay_tree_tag **container_category**
- typedef
  range_invalidation_guarantee **invalidation_guarantee**

4.206.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< splay_tree_tag >

Specialization, splay tree.

Definition at line 272 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.207 __gnu_pbds::container_traits_base< thin_heap_tag > Struct Template Reference

Public Types

- enum { **order_preserving**, **erase_can_throw**, **split_join_can_throw**, **reverse_iteration** }
- typedef thin_heap_tag **container_category**
- typedef
  point_invalidation_guarantee **invalidation_guarantee**

4.207.1 Detailed Description

template<>struct __gnu_pbds::container_traits_base< thin_heap_tag >

Specialization, thin heap.

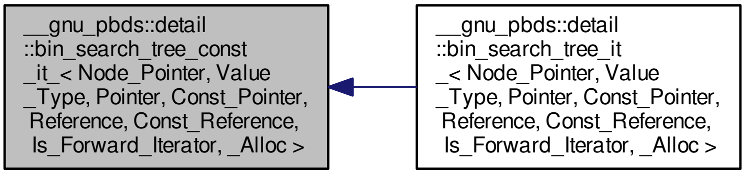
Definition at line 352 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.208 __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_const_it_** (const Node_Pointer p_nd=0)
- **bin_search_tree_const_it_** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other)
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other) const
- const_reference **operator∗** () const
- bin_search_tree_const_it_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- bin_search_tree_const_it_ < Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)

- bin_search_tree_const_it_
  < Node_Pointer, Value_Type,
  Pointer, Const_Pointer,
  Reference, Const_Reference,
  Is_Forward_Iterator, _Alloc > & **operator--** ()
- bin_search_tree_const_it_
  < Node_Pointer, Value_Type,
  Pointer, Const_Pointer,
  Reference, Const_Reference,
  Is_Forward_Iterator, _Alloc > **operator--** (int)
- const_pointer **operator->** () const
- bin_search_tree_const_it_
  < Node_Pointer, Value_Type,
  Pointer, Const_Pointer,
  Reference, Const_Reference,
  Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bin_search_tree_const_it_
  < Node_Pointer, Value_Type,
  Pointer, Const_Pointer,
  Reference, Const_Reference,
  Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

4.208.1 Detailed Description

template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

Const iterator.

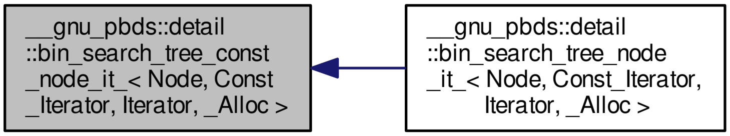
Definition at line 105 of file point_iterators.hpp.

The documentation for this class was generated from the following file:

- point_iterators.hpp

4.209 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Const_Iterator const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef _Alloc::template rebind< metadata_type > ::other::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef Const_Iterator reference
- typedef Const_Iterator value_type

Public Member Functions

- **bin_search_tree_const_node_it_** (const node_pointer p_nd=0)
- bin_search_tree_const_node_it_ < Node, Const_Iterator, Iterator, _Alloc > get_l_child () const
- metadata_const_reference get_metadata () const
- bin_search_tree_const_node_it_ < Node, Const_Iterator, Iterator, _Alloc > get_r_child () const
- bool operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const
- const_reference operator∗ () const
- bool operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

4.209.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Const node iterator.

Definition at line 58 of file bin_search_tree_/node_iterators.hpp.

4.209.2 Member Typedef Documentation

4.209.2.1 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::const_reference

Iterator's __const reference type.

Definition at line 80 of file bin_search_tree_/node_iterators.hpp.

4.209.2.2 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::difference_type

Difference type.

Definition at line 71 of file bin_search_tree_/node_iterators.hpp.

4.209.2.3 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::iterator_category

Category.

Definition at line 68 of file bin_search_tree_/node_iterators.hpp.

4.209.2.4 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference

Const metadata reference type.

Definition at line 88 of file bin_search_tree_/node_iterators.hpp.

4.209.2.5 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::metadata_type

Metadata type.

Definition at line 83 of file bin_search_tree_/node_iterators.hpp.

4.209.2.6 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference

Iterator's reference type.

Definition at line 77 of file bin_search_tree_/node_iterators.hpp.

4.209.2.7 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type

Iterator's value type.

Definition at line 74 of file bin_search_tree_/node_iterators.hpp.

4.209.3 Member Function Documentation

4.209.3.1 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_l_child ( ) const [inline]

Returns the __const node iterator associated with the left node.

Definition at line 107 of file bin_search_tree_/node_iterators.hpp.

4.209.3.2 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata ( ) const [inline]

Metadata access.

Definition at line 102 of file bin_search_tree_/node_iterators.hpp.

4.209.3.3 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_const_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_r_child ( ) const [inline]

Returns the __const node iterator associated with the right node.

Definition at line 112 of file bin_search_tree_/node_iterators.hpp.

4.209.3.4 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & *other*) const [inline]

Compares (negatively) to a different iterator object.

Definition at line 122 of file bin_search_tree_/node_iterators.hpp.

4.209.3.5 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* ( ) const [inline]

Access.

Definition at line 97 of file bin_search_tree_/node_iterators.hpp.

4.209.3.6 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & *other*) const [inline]

Compares to a different iterator object.

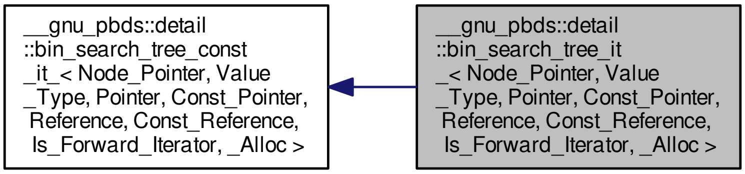
Definition at line 117 of file bin_search_tree_/node_iterators.hpp.

The documentation for this class was generated from the following file:

- bin_search_tree_/node_iterators.hpp

4.210 __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_it_** (const Node_Pointer p_nd=0)
- **bin_search_tree_it_** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other)
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other) const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::reference **operator*** () const
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()

- bin_search_tree_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc > **operator++** (int)
- bin_search_tree_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc > & **operator--** ()
- bin_search_tree_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc > **operator--** (int)
- bin_search_tree_const_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc >
::pointer **operator->** () const
- bin_search_tree_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bin_search_tree_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,!Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Types

- typedef
bin_search_tree_const_it_
< Node_Pointer, Value_Type,
Pointer, Const_Pointer,
Reference, Const_Reference,
Is_Forward_Iterator, _Alloc > **base_it_type**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

4.210.1 Detailed Description

template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

Iterator.

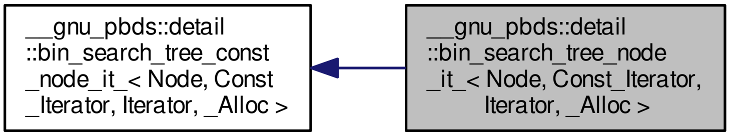
Definition at line 282 of file point_iterators.hpp.

The documentation for this class was generated from the following file:

- point_iterators.hpp

4.211 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Iterator const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef _Alloc::template rebind< metadata_type > ::other::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef Iterator reference
- typedef Iterator value_type

Public Member Functions

- **bin_search_tree_node_it_** (const node_pointer p_nd=0)
- bin_search_tree_node_it_< Node,
  Const_Iterator, Iterator,
  _Alloc > get_l_child () const
- metadata_const_reference get_metadata () const
- bin_search_tree_node_it_< Node,
  Const_Iterator, Iterator,
  _Alloc > get_r_child () const
- bool operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const
- Iterator operator∗ () const
- bool operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

4.211.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Node iterator.

Definition at line 136 of file bin_search_tree_/node_iterators.hpp.

4.211.2 Member Typedef Documentation

4.211.2.1 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::const_reference

Iterator's __const reference type.

Definition at line 153 of file bin_search_tree_/node_iterators.hpp.

4.211.2.2 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::difference_type `[inherited]`

Difference type.

Definition at line 71 of file bin_search_tree_/node_iterators.hpp.

4.211.2.3 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::iterator_category `[inherited]`

Category.

Definition at line 68 of file bin_search_tree_/node_iterators.hpp.

4.211.2.4 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type>::other::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]

Const metadata reference type.

Definition at line 88 of file bin_search_tree_/node_iterators.hpp.

4.211.2.5 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::metadata_type [inherited]

Metadata type.

Definition at line 83 of file bin_search_tree_/node_iterators.hpp.

4.211.2.6 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::reference

Iterator's reference type.

Definition at line 150 of file bin_search_tree_/node_iterators.hpp.

4.211.2.7 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::value_type

Iterator's value type.

Definition at line 147 of file bin_search_tree_/node_iterators.hpp.

4.211.3 Member Function Documentation

4.211.3.1 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_l_child ( ) const [inline]

Returns the node iterator associated with the left node.

Definition at line 167 of file bin_search_tree_/node_iterators.hpp.

4.211.3.2 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_metadata ( ) const [inline], [inherited]

Metadata access.

Definition at line 102 of file bin_search_tree_/node_iterators.hpp.

4.211.3.3 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bin_search_tree_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::get_r_child ( ) const [inline]

Returns the node iterator associated with the right node.

Definition at line 175 of file bin_search_tree_/node_iterators.hpp.

4.211.3.4 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & *other*) const `[inline]`, `[inherited]`

Compares (negatively) to a different iterator object.

Definition at line 122 of file bin_search_tree_/node_iterators.hpp.

4.211.3.5 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator* ( ) const `[inline]`

Access.

Definition at line 162 of file bin_search_tree_/node_iterators.hpp.

4.211.3.6 template<typename Node , class Const_Iterator , class Iterator , typename _Alloc > bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > & *other*) const `[inline]`, `[inherited]`

Compares to a different iterator object.

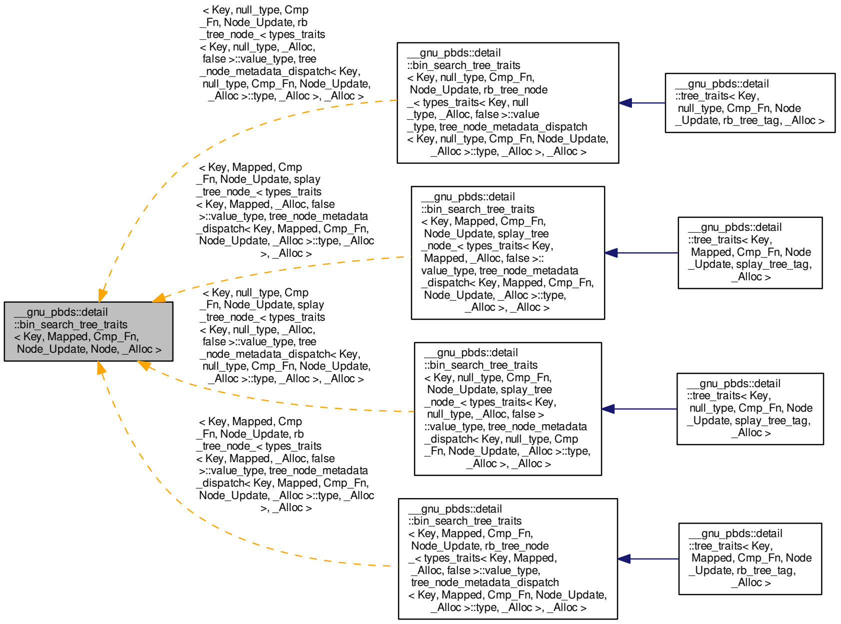
Definition at line 117 of file bin_search_tree_/node_iterators.hpp.

The documentation for this class was generated from the following file:

- bin_search_tree_/node_iterators.hpp

4.212 __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef Node **node**
- typedef bin_search_tree_const_node_it_ < Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef

bin_search_tree_node_it_< Node,
point_const_iterator,
point_iterator, _Alloc > **node_iterator**
- typedef Node_Update
< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef
__gnu_pbds::null_node_update
< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**
- typedef
bin_search_tree_const_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > **reverse_iterator**

4.212.1 Detailed Description

template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >

Binary search tree traits, primary template.

Definition at line 63 of file bin_search_tree_/traits.hpp.

4.212.2 Member Typedef Documentation

4.212.2.1 template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc> typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- bin_search_tree_/traits.hpp

4.213 __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference

Public Types

- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef Node **node**
- typedef bin_search_tree_const_node_it_ < Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update < node_const_iterator,

node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**
- typedef
bin_search_tree_const_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef const_reverse_iterator **reverse_iterator**

4.213.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node-_Update, class Node, typename _Alloc>struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

Specialization.

Definition at line 169 of file bin_search_tree_/traits.hpp.

4.213.2 Member Typedef Documentation

4.213.2.1 template<typename Key , class Cmp_Fn , template< typename Node_CItr, class Node_Itr, class Cmp_Fn, typename _Alloc > class Node_Update, class Node , typename _Alloc > typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

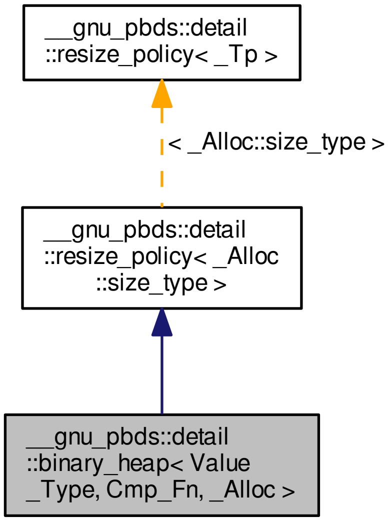
Definition at line 221 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- bin_search_tree_/traits.hpp

4.214 __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef cond_dealtor < value_type, _Alloc > **cond_dealtor_t**
- typedef binary_heap_const_iterator_ < value_type, entry, simple_value, _Alloc > **const_iterator**
- typedef value_allocator::const_pointer **const_pointer**
- typedef value_allocator::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef __conditional_type < simple_value, value_type, pointer >::__type **entry**
- typedef _Alloc::template rebind< entry >::other **entry_allocator**
- typedef entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple < Value_Type >::value >::type **entry_cmp**
- typedef entry_allocator::pointer **entry_pointer**

- typedef const_iterator **iterator**
- typedef
  binary_heap_point_const_iterator_
  < value_type, entry,
  simple_value, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef value_allocator::pointer **pointer**
- typedef value_allocator::reference **reference**
- typedef
  __gnu_pbds::detail::resize_policy
  < typename _Alloc::size_type > **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binary_heap** (const cmp_fn &)
- **binary_heap** (const binary_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- void **erase_at** (entry_pointer, size_type, false_type)
- void **erase_at** (entry_pointer, size_type, true_type)
- template<typename Pred >
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **join** (binary_heap &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- void **pop** ()
- point_iterator **push** (const_reference)
- bool **resize_needed_for_grow** (size_type) const
- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- size_type **size** () const
- template<typename Pred >
  void **split** (Pred, binary_heap &)
- void **swap** (resize_policy< _Alloc::size_type > &)
- void **swap** (binary_heap &)
- const_reference **top** () const

Static Public Attributes

- static const _Alloc::size_type **min_size**

Protected Member Functions

- template<typename It >
  void **copy_from_range** (It, It)

4.214.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >

Binary heaps composed of resize and compare policies.

Based on CLRS.

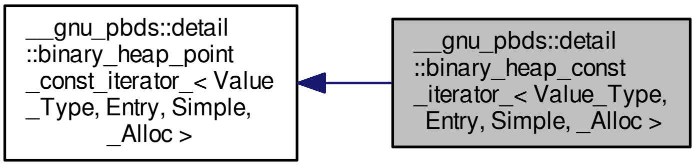
Definition at line 84 of file binary_heap_.hpp.

The documentation for this class was generated from the following files:

- binary_heap_.hpp
- binary_heap_/insert_fn_imps.hpp
- binary_heap_/constructors_destructor_fn_imps.hpp
- binary_heap_/iterators_fn_imps.hpp
- binary_heap_/erase_fn_imps.hpp
- binary_heap_/info_fn_imps.hpp
- binary_heap_/find_fn_imps.hpp
- binary_heap_/split_join_fn_imps.hpp
- binary_heap_/policy_access_fn_imps.hpp

4.215 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >:



Public Types

- typedef base_type::const_pointer const_pointer
- typedef base_type::const_reference const_reference
- typedef _Alloc::difference_type difference_type
- typedef std::forward_iterator_tag iterator_category
- typedef base_type::pointer pointer
- typedef base_type::reference reference
- typedef base_type::value_type value_type

Public Member Functions

- **binary_heap_const_iterator_** (entry_pointer p_e)
- binary_heap_const_iterator_ ()
- binary_heap_const_iterator_ (const binary_heap_const_iterator_ &other)
- bool operator!= (const binary_heap_const_iterator_ &other) const
- bool operator!= (const binary_heap_point_const_iterator_ &other) const
- const_reference operator∗ () const
- binary_heap_const_iterator_ & **operator++** ()
- binary_heap_const_iterator_ **operator++** (int)
- const_pointer operator-> () const
- bool operator== (const binary_heap_const_iterator_ &other) const
- bool operator== (const binary_heap_point_const_iterator_ &other) const

Public Attributes

- entry_pointer **m_p_e**

4.215.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

Definition at line 60 of file binary_heap_/const_iterator.hpp.

4.215.2 Member Typedef Documentation

4.215.2.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_pointer

Iterator's const pointer type.

Definition at line 80 of file binary_heap_/const_iterator.hpp.

4.215.2.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_reference

Iterator's const reference type.

Definition at line 86 of file binary_heap_/const_iterator.hpp.

4.215.2.3 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::difference_type

Difference type.

Definition at line 71 of file binary_heap_/const_iterator.hpp.

4.215.2.4 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::iterator_category

Category.

Definition at line 68 of file binary_heap_/const_iterator.hpp.

4.215.2.5 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::pointer

Iterator's pointer type.

Definition at line 77 of file binary_heap_/const_iterator.hpp.

4.215.2.6 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::reference

Iterator's reference type.

Definition at line 83 of file binary_heap_/const_iterator.hpp.

4.215.2.7 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef base_type::value_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type

Iterator's value type.

Definition at line 74 of file binary_heap_/const_iterator.hpp.

4.215.3 Constructor & Destructor Documentation

4.215.3.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_ ( ) `[inline]`

Default constructor.

Definition at line 94 of file binary_heap_/const_iterator.hpp.

4.215.3.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_ (const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) `[inline]`

Copy constructor.

Definition at line 99 of file binary_heap_/const_iterator.hpp.

4.215.4 Member Function Documentation

4.215.4.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!= (const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) const `[inline]`

Compares content (negatively) to a different iterator object.

Definition at line 110 of file binary_heap_/const_iterator.hpp.

4.215.4.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!= (const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) const `[inline]`, `[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 126 of file binary_heap_/point_const_iterator.hpp.

4.215.4.3 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator* ( ) const `[inline]`, `[inherited]`

Access.

Definition at line 113 of file binary_heap_/point_const_iterator.hpp.

4.215.4.4 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator-> ( ) const `[inline]`, `[inherited]`

Access.

Definition at line 105 of file binary_heap_/point_const_iterator.hpp.

4.215.4.5 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== (const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) const `[inline]`

Compares content to a different iterator object.

Definition at line 105 of file binary_heap_/const_iterator.hpp.

4.215.4.6 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== (const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) const `[inline]`, `[inherited]`

Compares content to a different iterator object.

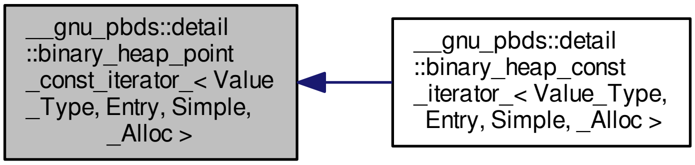
Definition at line 121 of file binary_heap_/point_const_iterator.hpp.

The documentation for this class was generated from the following file:

- binary_heap_/const_iterator.hpp

4.216 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >:



Public Types

- typedef _Alloc::template rebind< value_type > ::other::const_pointer const_pointer
- typedef _Alloc::template rebind< value_type > ::other::const_reference const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef _Alloc::template rebind< value_type > ::other::pointer pointer
- typedef _Alloc::template rebind< value_type > ::other::reference reference
- typedef Value_Type value_type

Public Member Functions

- **binary_heap_point_const_iterator_** (entry_pointer p_e)
- binary_heap_point_const_iterator_ ()
- binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_ &other)
- bool operator!= (const binary_heap_point_const_iterator_ &other) const
- const_reference operator* () const
- const_pointer operator-> () const
- bool operator== (const binary_heap_point_const_iterator_ &other) const

Public Attributes

- entry_pointer **m_p_e**

Protected Types

- typedef _Alloc::template
  rebind< Entry >
  ::other::pointer **entry_pointer**

4.216.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

Definition at line 55 of file binary_heap_/point_const_iterator.hpp.

4.216.2 Member Typedef Documentation

4.216.2.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template rebind<value_type>::other::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_pointer

Iterator's const pointer type.

Definition at line 77 of file binary_heap_/point_const_iterator.hpp.

4.216.2.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template rebind<value_type>::other::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::const_reference

Iterator's const reference type.

Definition at line 87 of file binary_heap_/point_const_iterator.hpp.

4.216.2.3 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::difference_type

Difference type.

Definition at line 65 of file binary_heap_/point_const_iterator.hpp.

4.216.2.4 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef trivial_iterator_tag __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::iterator_category

Category.

Definition at line 62 of file binary_heap_/point_const_iterator.hpp.

4.216.2.5 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template rebind<value_type>::other::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::pointer

Iterator's pointer type.

Definition at line 72 of file binary_heap_/point_const_iterator.hpp.

4.216.2.6 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef _Alloc::template rebind<value_type>::other::reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::reference

Iterator's reference type.

Definition at line 82 of file binary_heap_/point_const_iterator.hpp.

4.216.2.7 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > typedef Value_Type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::value_type

Iterator's value type.

Definition at line 68 of file binary_heap_/point_const_iterator.hpp.

4.216.3 Constructor & Destructor Documentation

4.216.3.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_ ( ) [inline]

Default constructor.

Definition at line 95 of file binary_heap_/point_const_iterator.hpp.

4.216.3.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) [inline]

Copy constructor.

Definition at line 99 of file binary_heap_/point_const_iterator.hpp.

4.216.4 Member Function Documentation

4.216.4.1 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!= (const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 126 of file binary_heap_/point_const_iterator.hpp.

4.216.4.2 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator* ( ) const [inline]

Access.

Definition at line 113 of file binary_heap_/point_const_iterator.hpp.

4.216.4.3 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator-> ( ) const [inline]

Access.

Definition at line 105 of file binary_heap_/point_const_iterator.hpp.

4.216.4.4 template<typename Value_Type , typename Entry , bool Simple, typename _Alloc > bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator== (const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & *other*) const [inline]

Compares content to a different iterator object.

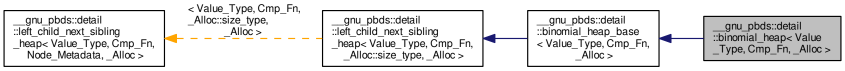
Definition at line 121 of file binary_heap_/point_const_iterator.hpp.

The documentation for this class was generated from the following file:

- binary_heap_/point_const_iterator.hpp

4.217 __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binomial_heap** (const Cmp_Fn &)
- **binomial_heap** (const binomial_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)

- template<typename Pred >
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template
  rebind
  < left_child_next_sibling_heap_node_
  < Value_Type,
  _Alloc::size_type, _Alloc >
  >::other **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair
  < node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.217.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Binomial heap.

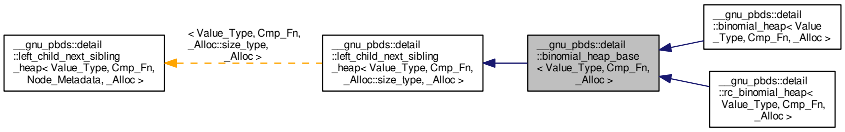
Definition at line 68 of file binomial_heap_.hpp.

The documentation for this class was generated from the following files:

- binomial_heap_.hpp
- binomial_heap_/constructors_destructor_fn_imps.hpp

4.218 __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template
  rebind
  < left_child_next_sibling_heap_node_
  < Value_Type,
  _Alloc::size_type, _Alloc >
  >::other **node_allocator**
- typedef
  base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef std::pair
  < node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **binomial_heap_base** (const Cmp_Fn &)
- **binomial_heap_base** (const binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)

- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.218.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >

Base class for binomial heap.

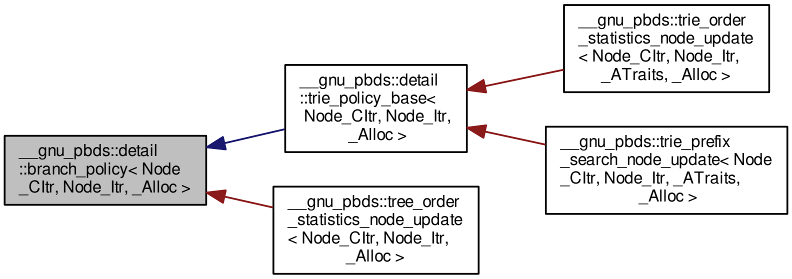
Definition at line 77 of file binomial_heap_base_.hpp.

The documentation for this class was generated from the following files:

- binomial_heap_base_.hpp
- binomial_heap_base_/constructors_destructor_fn_imps.hpp
- binomial_heap_base_/find_fn_imps.hpp
- binomial_heap_base_/insert_fn_imps.hpp
- binomial_heap_base_/erase_fn_imps.hpp
- binomial_heap_base_/split_join_fn_imps.hpp

4.219 __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >:



Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef rebind_k::const_reference **key_const_reference**
- typedef value_type::first_type **key_type**
- typedef remove_const< key_type >
  ::type **rckey_type**
- typedef remove_const
  < value_type >::type **rcvalue_type**
- typedef _Alloc::template
  rebind< rckey_type >::other **rebind_k**
- typedef _Alloc::template
  rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits
  < it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** ()=0
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

4.219.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _Alloc>struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >

Primary template, base class for branch structure policies.

Definition at line 52 of file branch_policy.hpp.

The documentation for this struct was generated from the following file:

- branch_policy.hpp

4.220 __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc > Struct Template Reference

Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_CItr::value_type **it_type**
- typedef rebind_v::const_reference **key_const_reference**
- typedef value_type **key_type**

- typedef remove_const < value_type >::type **rcvalue_type**
- typedef _Alloc::template rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits < it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** () const =0
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

4.220.1 Detailed Description

template<typename Node_CItr, typename _Alloc>struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >

Specialization for const iterators.

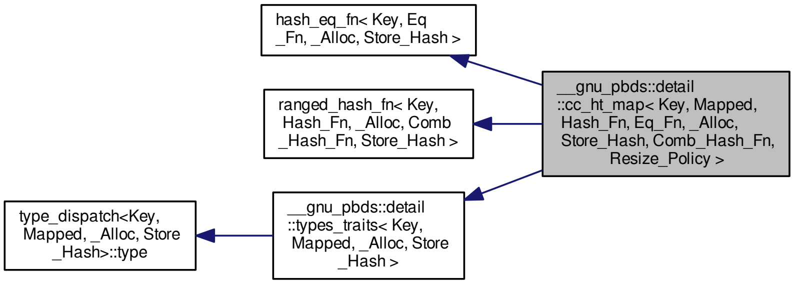
Definition at line 88 of file branch_policy.hpp.

The documentation for this struct was generated from the following file:

- branch_policy.hpp

4.221 __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **cc_ht_map** (const Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &, const Resize_Policy &)
- **cc_ht_map** (const cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()

- template<typename It >
  void **copy_from_range** (It, It)
- bool empty () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred >
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Hash_Fn & get_comb_hash_fn ()
- const Comb_Hash_Fn & get_comb_hash_fn () const
- Eq_Fn & get_eq_fn ()
- const Eq_Fn & get_eq_fn () const
- Hash_Fn & get_hash_fn ()
- const Hash_Fn & get_hash_fn () const
- Resize_Policy & get_resize_policy ()
- const Resize_Policy & get_resize_policy () const
- void **initialize** ()
- std::pair< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

4.221.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

A collision-chaining hash-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor. Default is __gnu_cxx::hash.
Eq_Fn	Equal functor. Default std::equal_to<Key>
_Alloc	Allocator type.
Store_Hash	If key type stores extra metadata. Defaults to false.
Comb_Hash_Fn	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::-Hash Policies.) Default direct_mask_range_hashing.
Resize_Policy	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_-policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_hash_fn, detail::types_traits. (Optional: detail::debug_-map_base.)

Definition at line 139 of file cc_ht_map_.hpp.

4.221.2 Member Enumeration Documentation

4.221.2.1 template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy> anonymous enum

Value stores hash, true or false.

Definition at line 200 of file cc_ht_map_.hpp.

4.221.3 Member Function Documentation

4.221.3.1 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::empty ( ) const [inline]

True if size() == 0.

Definition at line 52 of file cc_ht_map_.hpp.

4.221.3.2 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ( )

Return current comb_hash_fn.

Definition at line 70 of file cc_ht_map_.hpp.

4.221.3.3 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ( ) const

Return current const comb_hash_fn.

Definition at line 76 of file cc_ht_map_.hpp.

4.221.3.4 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ( )

Return current eq_fn.

Definition at line 58 of file cc_ht_map_.hpp.

4.221.3.5 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ( ) const

Return current const eq_fn.

Definition at line 64 of file cc_ht_map_.hpp.

4.221.3.6 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn ( )

Return current hash_fn.

Definition at line 46 of file cc_ht_map_.hpp.

4.221.3.7 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_hash_fn ( ) const

Return current const hash_fn.

Definition at line 52 of file cc_ht_map_.hpp.

4.221.3.8 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ( )

Return current resize_policy.

Definition at line 82 of file cc_ht_map_.hpp.

4.221.3.9 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy ( ) const

Return current const resize_policy.

Definition at line 88 of file cc_ht_map_.hpp.

The documentation for this class was generated from the following files:

- cc_ht_map_.hpp
- cc_hash_table_map_/constructor_destructor_fn_imps.hpp
- entry_list_fn_imps.hpp
- cc_hash_table_map_/find_fn_imps.hpp
- cc_hash_table_map_/resize_fn_imps.hpp
- cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp
- cc_hash_table_map_/resize_store_hash_fn_imps.hpp
- size_fn_imps.hpp
- cc_hash_table_map_/policy_access_fn_imps.hpp

- cc_hash_table_map_/erase_fn_imps.hpp
- cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp
- cc_hash_table_map_/erase_store_hash_fn_imps.hpp
- cc_hash_table_map_/iterators_fn_imps.hpp
- cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp
- cc_hash_table_map_/insert_store_hash_fn_imps.hpp

4.222 __gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference

Public Types

- typedef HT_Map::entry **entry**
- typedef HT_Map::entry_allocator **entry_allocator**
- typedef __rebind_e::other **entry_allocator**
- typedef entry_allocator::pointer **entry_pointer**
- typedef HT_Map::key_type **key_type**

Public Member Functions

- **cond_dealtor** (entry_allocator ∗p_a, entry ∗p_e)
- **cond_dealtor** (entry_pointer p_e)
- void **set_key_destruct** ()
- void **set_no_action** ()
- void **set_no_action_destructor** ()

Protected Attributes

- bool **m_key_destruct**
- entry_allocator ∗const **m_p_a**
- entry ∗const **m_p_e**

4.222.1 Detailed Description

template<typename Entry, typename _Alloc>class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Conditional deallocate constructor argument.

Conditional dey destructor, cc_hash.

Definition at line 50 of file cond_dealtor.hpp.

The documentation for this class was generated from the following files:

- cond_dealtor.hpp
- cond_key_dtor_entry_dealtor.hpp

4.223 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference

Public Types

- typedef binary_heap< _VTp,
  Cmp_Fn, _Alloc > type

4.223.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >

Specialization for binary_heap.

Definition at line 95 of file priority_queue_base_dispatch.hpp.

4.223.2 Member Typedef Documentation

4.223.2.1 template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >::type

Dispatched type.

Definition at line 99 of file priority_queue_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

4.224 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference

Public Types

- typedef binomial_heap< _VTp,
  Cmp_Fn, _Alloc > type

4.224.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >

Specialization for binomial_heap.

Definition at line 77 of file priority_queue_base_dispatch.hpp.

4.224.2 Member Typedef Documentation

4.224.2.1 template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >::type

Dispatched type.

Definition at line 81 of file priority_queue_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

4.225 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference

Public Types

- typedef pairing_heap< _VTp,
  Cmp_Fn, _Alloc > type

4.225.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >

Specialization for pairing_heap.

Definition at line 68 of file priority_queue_base_dispatch.hpp.

4.225.2 Member Typedef Documentation

4.225.2.1 template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >::type

Dispatched type.

Definition at line 72 of file priority_queue_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

4.226 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference

Public Types

- typedef rc_binomial_heap< _VTp,
  Cmp_Fn, _Alloc > type

4.226.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >

Specialization for rc_binary_heap.

Definition at line 86 of file priority_queue_base_dispatch.hpp.

4.226.2 Member Typedef Documentation

4.226.2.1 template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type

Dispatched type.

Definition at line 90 of file priority_queue_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

4.227 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference

Public Types

- typedef thin_heap< _VTp,
  Cmp_Fn, _Alloc > type

4.227.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >

Specialization for thin_heap.

Definition at line 104 of file priority_queue_base_dispatch.hpp.

4.227.2 Member Typedef Documentation

4.227.2.1 template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >::type

Dispatched type.

Definition at line 108 of file priority_queue_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

4.228 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef cc_ht_map< Key, Mapped,
  at0t, at1t, _Alloc,
  at3t::value, at4t, at2t > type

4.228.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >

Specialization colision-chaining hash map.

Definition at line 258 of file container_base_dispatch.hpp.

4.228.2 Member Typedef Documentation

4.228.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >::type

Dispatched type.

Definition at line 275 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.229 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef gp_ht_map< Key, Mapped,
  at0t, at1t, _Alloc,
  at3t::value, at4t, at5t, at2t > type

4.229.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >

Specialization general-probe hash map.

Definition at line 303 of file container_base_dispatch.hpp.

4.229.2 Member Typedef Documentation

4.229.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type

Dispatched type.

Definition at line 322 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.230 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef lu_map< Key, Mapped, at0t, _Alloc, at1t > type

4.230.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >

Specialization for list-update map.

Definition at line 107 of file container_base_dispatch.hpp.

4.230.2 Member Typedef Documentation

4.230.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >::type

Dispatched type.

Definition at line 118 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.231 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef ov_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

4.231.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >

Specialization ordered-vector tree map.

Definition at line 227 of file container_base_dispatch.hpp.

4.231.2 Member Typedef Documentation

4.231.2.1 template< typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type

Dispatched type.

Definition at line 237 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.232 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef pat_trie_map< Key,
  Mapped, at1t, _Alloc > **type**

4.232.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >

Specialization for PATRICIA trie map.

Definition at line 139 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.233 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef rb_tree_map< Key,
  Mapped, at0t, at1t, _Alloc > type

4.233.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >

Specialization for R-B tree map.

Definition at line 165 of file container_base_dispatch.hpp.

4.233.2 Member Typedef Documentation

4.233.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef **rb_tree_map**<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, **rb_tree_tag**, Policy_Tl >::**type**

Dispatched type.

Definition at line 175 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.234 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef splay_tree_map< Key,
  Mapped, at0t, at1t, _Alloc > type

4.234.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >

Specialization splay tree map.

Definition at line 195 of file container_base_dispatch.hpp.

4.234.2 Member Typedef Documentation

4.234.2.1 template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl > typedef **splay_tree_map**<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, **splay_tree_tag**, Policy_Tl >::**type**

Dispatched type.

Definition at line 206 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.235 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef cc_ht_set< Key,
  null_type, at0t, at1t, _Alloc,
  at3t::value, at4t, at2t > type

4.235.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >

Specialization colision-chaining hash set.

Definition at line 280 of file container_base_dispatch.hpp.

4.235.2 Member Typedef Documentation

4.235.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >::type

Dispatched type.

Definition at line 298 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.236 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef gp_ht_set< Key,
  null_type, at0t, at1t, _Alloc,
  at3t::value, at4t, at5t, at2t > type

4.236.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >

Specialization general-probe hash set.

Definition at line 327 of file container_base_dispatch.hpp.

4.236.2 Member Typedef Documentation

4.236.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type

Dispatched type.

Definition at line 347 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.237 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef lu_set< Key, null_type,
  at0t, _Alloc, at1t > type

4.237.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >

Specialization for list-update set.

Definition at line 123 of file container_base_dispatch.hpp.

4.237.2 Member Typedef Documentation

4.237.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type

Dispatched type.

Definition at line 134 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.238 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef ov_tree_set< Key,
  null_type, at0t, at1t, _Alloc > type

4.238.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >

Specialization ordered-vector tree set.

Definition at line 242 of file container_base_dispatch.hpp.

4.238.2 Member Typedef Documentation

4.238.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type

Dispatched type.

Definition at line 253 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.239 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef pat_trie_set< Key,
  null_type, at1t, _Alloc > type

4.239.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >

Specialization for PATRICIA trie set.

Definition at line 151 of file container_base_dispatch.hpp.

4.239.2 Member Typedef Documentation

4.239.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type

Dispatched type.

Definition at line 160 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.240 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef rb_tree_set< Key,
  null_type, at0t, at1t, _Alloc > **type**

4.240.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >

Specialization for R-B tree set.

Definition at line 180 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.241 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference

Public Types

- typedef splay_tree_set< Key, null_type, at0t, at1t, _Alloc > type

4.241.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >

Specialization splay tree set.

Definition at line 211 of file container_base_dispatch.hpp.

4.241.2 Member Typedef Documentation

4.241.2.1 template<typename Key , typename _Alloc , typename Policy_Tl > typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type

Dispatched type.

Definition at line 222 of file container_base_dispatch.hpp.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

4.242 __gnu_pbds::detail::default_comb_hash_fn Struct Reference

Public Types

- typedef direct_mask_range_hashing type

4.242.1 Detailed Description

Primary template, default_comb_hash_fn.

Definition at line 80 of file standard_policies.hpp.

4.242.2 Member Typedef Documentation

4.242.2.1 typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type

Dispatched type.

Definition at line 83 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.243 __gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference

Public Types

- typedef std::equal_to< Key > type

4.243.1 Detailed Description

template<typename Key>struct __gnu_pbds::detail::default_eq_fn< Key >

Primary template, default_eq_fn.

Definition at line 67 of file standard_policies.hpp.

4.243.2 Member Typedef Documentation

4.243.2.1 template<typename Key> typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type

Dispatched type.

Definition at line 70 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.244 __gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference

Public Types

- typedef std::tr1::hash< Key > type

4.244.1 Detailed Description

template<typename Key>struct __gnu_pbds::detail::default_hash_fn< Key >

Primary template, default_hash_fn.

Definition at line 59 of file standard_policies.hpp.

4.244.2 Member Typedef Documentation

4.244.2.1 template<typename Key> typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type

Dispatched type.

Definition at line 62 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.245 __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference

Public Types

- typedef cond_type::__type type

4.245.1 Detailed Description

template<typename Comb_Probe_Fn>struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >

Primary template, default_probe_fn.

Definition at line 117 of file standard_policies.hpp.

4.245.2 Member Typedef Documentation

4.245.2.1 template<typename Comb_Probe_Fn > typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type

Dispatched type.

Definition at line 129 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.246 __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference

Public Types

- typedef
  hash_standard_resize_policy
  < size_policy_type, trigger,
  false, size_type > type

4.246.1 Detailed Description

template<typename Comb_Hash_Fn>struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >

Primary template, default_resize_policy.

Definition at line 88 of file standard_policies.hpp.

4.246.2 Member Typedef Documentation

4.246.2.1 template<typename Comb_Hash_Fn> typedef **hash_standard_resize_policy**<size_policy_type, **trigger**, false, size_type> **__gnu_pbds::detail::default_resize_policy**< Comb_Hash_Fn >**::type**

Dispatched type.

Definition at line 105 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.247 __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference

Public Types

- typedef
  trie_string_access_traits
  < string_type > type

4.247.1 Detailed Description

template<typename Char, typename Char_Traits>struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >

Partial specialization, default_trie_access_traits.

Definition at line 142 of file standard_policies.hpp.

4.247.2 Member Typedef Documentation

4.247.2.1 template<typename Char , typename Char_Traits > typedef **trie_string_access_traits**<**string_type**> __gnu_pbds::detail::default_trie_access_traits< **std::basic_string**< Char, Char_Traits, **std::allocator**< char > > >**::type**

Dispatched type.

Definition at line 149 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.248 __gnu_pbds::detail::default_update_policy Struct Reference

Public Types

- typedef lu_move_to_front_policy type

4.248.1 Detailed Description

Default update policy.

Definition at line 109 of file standard_policies.hpp.

4.248.2 Member Typedef Documentation

4.248.2.1 typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type

Dispatched type.

Definition at line 112 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.249 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference

Public Types

- typedef const_iterator **const_reference**
- typedef const_reference **reference**
- typedef const_iterator **value_type**

4.249.1 Detailed Description

template<typename Key, typename Data, typename _Alloc>struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >

Constant node iterator.

Definition at line 52 of file null_node_metadata.hpp.

The documentation for this struct was generated from the following file:

- null_node_metadata.hpp

4.250 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false > Struct Template Reference

Classes

- struct type

  Compare plus entry.

Public Types

- typedef
  __rebind_v::other::const_pointer **entry**

4.250.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >

Specialization, false.

Definition at line 62 of file entry_cmp.hpp.

The documentation for this struct was generated from the following file:

- entry_cmp.hpp

4.251 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference

Inherits Cmp_Fn.

Public Member Functions

- **type** (const Cmp_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const

4.251.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type

Compare plus entry.

Definition at line 71 of file entry_cmp.hpp.

The documentation for this struct was generated from the following file:

- entry_cmp.hpp

4.252 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true > Struct Template Reference

Public Types

- typedef Cmp_Fn type

4.252.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >

Specialization, true.

Definition at line 54 of file entry_cmp.hpp.

4.252.2 Member Typedef Documentation

4.252.2.1 template<typename _VTp , typename Cmp_Fn , typename _Alloc > typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type

Compare.

Definition at line 57 of file entry_cmp.hpp.

The documentation for this struct was generated from the following file:

- entry_cmp.hpp

4.253 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false > Struct Template Reference

Public Types

- typedef
  __rebind_v::other::const_pointer **entry**

4.253.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc>struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >

Specialization, false.

Definition at line 61 of file entry_pred.hpp.

The documentation for this struct was generated from the following file:

- entry_pred.hpp

4.254 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true > Struct Template Reference

Public Types

- typedef Pred **type**

4.254.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc>struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >

Specialization, true.

Definition at line 54 of file entry_pred.hpp.

The documentation for this struct was generated from the following file:

- entry_pred.hpp

4.255 __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn > Struct Template Reference

Inherits Cmp_Fn.

Public Member Functions

- bool **operator()** (const Key &r_lhs, const Key &r_rhs) const

4.255.1 Detailed Description

template<typename Key, class Cmp_Fn>struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >

Equivalence function.

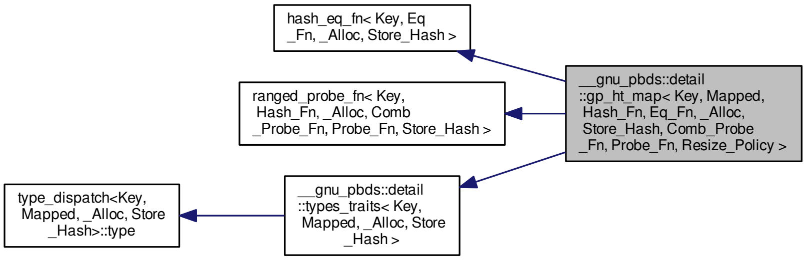
Definition at line 56 of file eq_by_less.hpp.

The documentation for this struct was generated from the following file:

- eq_by_less.hpp

4.256 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**

- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef Probe_Fn **probe_fn**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **gp_ht_map** (const gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)
- **gp_ht_map** (const Hash_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &, const Resize_Policy &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It > void **copy_from_range** (It, It)
- bool empty () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred > size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Probe_Fn & get_comb_probe_fn ()

- const Comb_Probe_Fn & get_comb_probe_fn () const
- Eq_Fn & get_eq_fn ()
- const Eq_Fn & get_eq_fn () const
- Hash_Fn & get_hash_fn ()
- const Hash_Fn & get_hash_fn () const
- Probe_Fn & get_probe_fn ()
- const Probe_Fn & get_probe_fn () const
- Resize_Policy & get_resize_policy ()
- const Resize_Policy & get_resize_policy () const
- std::pair< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

4.256.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

A general-probing hash-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor. Default is __gnu_cxx::hash.
Eq_Fn	Equal functor. Default std::equal_to<Key>
_Alloc	Allocator type.
Store_Hash	If key type stores extra metadata. Defaults to false.
Comb_Probe_Fn	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default direct_mask_range_hashing.
Probe_Fn	Probe functor. Defaults to linear_probe_fn, also quadratic_probe_fn.
Resize_Policy	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_probe_fn, detail::types_traits. (Optional: detail::debug_map_base.)

Definition at line 142 of file gp_ht_map_.hpp.

4.256.2 Member Enumeration Documentation

4.256.2.1 template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy> anonymous enum

Value stores hash, true or false.

Definition at line 208 of file gp_ht_map_.hpp.

4.256.3 Member Function Documentation

4.256.3.1 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::empty ( ) const `[inline]`

True if size() == 0.

Definition at line 58 of file gp_ht_map_.hpp.

4.256.3.2 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ( )

Return current comb_probe_fn.

Definition at line 82 of file gp_ht_map_.hpp.

4.256.3.3 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ( ) const

Return current const comb_probe_fn.

Definition at line 88 of file gp_ht_map_.hpp.

4.256.3.4 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ( )

Return current eq_fn.

Definition at line 58 of file gp_ht_map_.hpp.

4.256.3.5 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ( ) const

Return current const eq_fn.

Definition at line 64 of file gp_ht_map_.hpp.

4.256.3.6 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ( )

Return current hash_fn.

Definition at line 46 of file gp_ht_map_.hpp.

4.256.3.7 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ( ) const

Return current const hash_fn.

Definition at line 52 of file gp_ht_map_.hpp.

4.256.3.8 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ( )

Return current probe_fn.

Definition at line 70 of file gp_ht_map_.hpp.

4.256.3.9 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ( ) const

Return current const probe_fn.

Definition at line 76 of file gp_ht_map_.hpp.

4.256.3.10 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ( )

Return current resize_policy.

Definition at line 94 of file gp_ht_map_.hpp.

4.256.3.11 template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc , bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy > const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ( ) const

Return current const resize_policy.

Definition at line 100 of file gp_ht_map_.hpp.

The documentation for this class was generated from the following files:

- gp_ht_map_.hpp
- gp_hash_table_map_/constructor_destructor_fn_imps.hpp
- gp_hash_table_map_/find_fn_imps.hpp

- gp_hash_table_map_/resize_fn_imps.hpp
- gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp
- gp_hash_table_map_/resize_store_hash_fn_imps.hpp
- gp_hash_table_map_/info_fn_imps.hpp
- gp_hash_table_map_/policy_access_fn_imps.hpp
- gp_hash_table_map_/erase_fn_imps.hpp
- gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp
- gp_hash_table_map_/erase_store_hash_fn_imps.hpp
- iterator_fn_imps.hpp
- gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp
- gp_hash_table_map_/insert_store_hash_fn_imps.hpp

4.257 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference

Inherits Eq_Fn.

Public Types

- typedef Eq_Fn **eq_fn_base**
- typedef _Alloc::template
  rebind< Key >::other **key_allocator**
- typedef
  key_allocator::const_reference **key_const_reference**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, key_const_reference r_rhs_key) const
- void **swap** (const hash_eq_fn &other)

4.257.1 Detailed Description

template<typename Key, typename Eq_Fn, typename _Alloc>struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >

Specialization 1 - The client requests that hash values not be stored.

Definition at line 58 of file hash_eq_fn.hpp.

The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

4.258 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference

Inherits Eq_Fn.

Public Types

- typedef Eq_Fn **eq_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, size_type lhs_hash, key_const_reference r_rhs_key, size_type rhs_hash) const
- void **swap** (const hash_eq_fn &other)

4.258.1 Detailed Description

template<typename Key, class Eq_Fn, class _Alloc>struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >

Specialization 2 - The client requests that hash values be stored.

Definition at line 81 of file hash_eq_fn.hpp.

The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

4.259 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference

Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- size_type **get_size** () const
- void **set_size** (size_type size)
- void **swap** (hash_load_check_resize_trigger_size_base &other)

4.259.1 Detailed Description

template<typename Size_Type>class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >

Specializations.

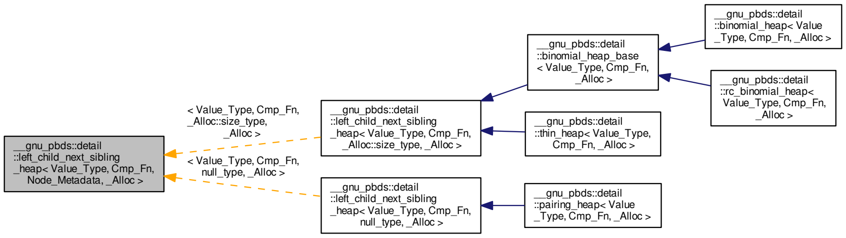
Definition at line 54 of file hash_load_check_resize_trigger_size_base.hpp.

The documentation for this class was generated from the following file:

- hash_load_check_resize_trigger_size_base.hpp

4.260 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef left_child_next_sibling_heap_const_iterator_ < node, _Alloc > **const_iterator**
- typedef __rebind_v::other::const_pointer **const_pointer**
- typedef __rebind_v::other::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef const_iterator **iterator**
- typedef left_child_next_sibling_heap_node_point_const_iterator_ < node, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef __rebind_v::other::pointer **pointer**
- typedef __rebind_v::other::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **left_child_next_sibling_heap** (const Cmp_Fn &)
- **left_child_next_sibling_heap** (const left_child_next_sibling_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const

- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **max_size** () const
- size_type **size** () const
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &)

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template rebind < left_child_next_sibling_heap_node_ < Value_Type, Node_Metadata, _Alloc > >::other **node_allocator**
- typedef node_allocator::const_pointer **node_const_pointer**
- typedef Node_Metadata **node_metadata**
- typedef node_allocator::pointer **node_pointer**
- typedef std::pair < node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- node_pointer **get_new_node_for_insert** (const_reference)
- template<typename Pred > node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.260.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

Base class for a basic heap.

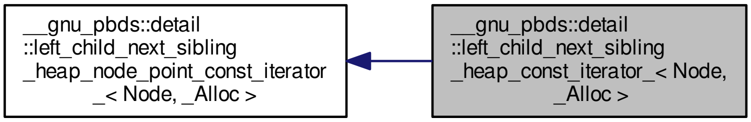
Definition at line 90 of file left_child_next_sibling_heap_.hpp.

The documentation for this class was generated from the following files:

- left_child_next_sibling_heap_.hpp
- left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp
- left_child_next_sibling_heap_/iterators_fn_imps.hpp
- left_child_next_sibling_heap_/insert_fn_imps.hpp
- left_child_next_sibling_heap_/erase_fn_imps.hpp
- left_child_next_sibling_heap_/info_fn_imps.hpp
- left_child_next_sibling_heap_/policy_access_fn_imps.hpp

4.261 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >:



Public Types

- typedef base_type::const_pointer const_pointer
- typedef base_type::const_reference const_reference
- typedef _Alloc::difference_type difference_type
- typedef std::forward_iterator_tag iterator_category
- typedef base_type::pointer pointer
- typedef base_type::reference reference
- typedef base_type::value_type value_type

Public Member Functions

- **left_child_next_sibling_heap_const_iterator_** (node_pointer p_nd)
- left_child_next_sibling_heap_const_iterator_ ()
- left_child_next_sibling_heap_const_iterator_ (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other)
- bool operator!= (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const
- bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const
- const_reference operator∗ () const
- left_child_next_sibling_heap_const_iterator_ < Node, _Alloc > & **operator++** ()
- left_child_next_sibling_heap_const_iterator_ < Node, _Alloc > **operator++** (int)

- const_pointer operator-> () const
- bool operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const
- bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

4.261.1 Detailed Description

template<typename Node, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >

Const point-type iterator.

Definition at line 60 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2 Member Typedef Documentation

4.261.2.1 template<typename Node , typename _Alloc > typedef base_type::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_pointer

Iterator's const pointer type.

Definition at line 81 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2.2 template<typename Node , typename _Alloc > typedef base_type::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::const_reference

Iterator's const reference type.

Definition at line 87 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2.3 template<typename Node , typename _Alloc > typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::difference_type

Difference type.

Definition at line 72 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2.4 template<typename Node , typename _Alloc > typedef std::forward_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::iterator_category

Category.

Definition at line 69 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2.5 template<typename Node , typename _Alloc > typedef base_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::pointer

Iterator's pointer type.

Definition at line 78 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2.6 template<typename Node , typename _Alloc > typedef base_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::reference

Iterator's reference type.

Definition at line 84 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.2.7 template<typename Node , typename _Alloc > typedef base_type::value_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::value_type

Iterator's value type.

Definition at line 75 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.3 Constructor & Destructor Documentation

4.261.3.1 template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ ( ) [inline]

Default constructor.

Definition at line 96 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.3.2 template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_const_iterator_ (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & *other*) [inline]

Copy constructor.

Definition at line 101 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.4 Member Function Documentation

4.261.4.1 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!= (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & *other*) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 111 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.4.2 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & *other*) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 137 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.261.4.3 template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator* ( ) const [inline], [inherited]

Access.

Definition at line 124 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.261.4.4 template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> ( ) const [inline], [inherited]

Access.

Definition at line 116 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.261.4.5 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & *other*) const [inline]

Compares content to a different iterator object.

Definition at line 106 of file left_child_next_sibling_heap_/const_iterator.hpp.

4.261.4.6 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & *other*) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 132 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_/const_iterator.hpp

4.262 __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference

Public Types

- typedef _Metadata **metadata_type**
- typedef _Alloc::template rebind< this_type > ::other::pointer **node_pointer**
- typedef _Alloc::size_type **size_type**
- typedef _Value **value_type**

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_l_child**
- node_pointer **m_p_next_sibling**
- node_pointer **m_p_prev_or_parent**
- value_type **m_value**

4.262.1 Detailed Description

template<typename _Value, typename _Metadata, typename _Alloc>struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >

Node.

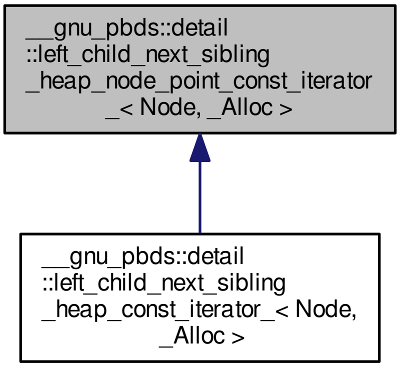
Definition at line 50 of file left_child_next_sibling_heap_/node.hpp.

The documentation for this struct was generated from the following file:

- left_child_next_sibling_heap_/node.hpp

4.263 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >:



Public Types

- typedef _Alloc::template rebind< value_type > ::other::const_pointer const_pointer
- typedef _Alloc::template rebind< value_type > ::other::const_reference const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef _Alloc::template rebind< value_type > ::other::pointer pointer
- typedef _Alloc::template rebind< value_type > ::other::reference reference
- typedef Node::value_type value_type

Public Member Functions

- **left_child_next_sibling_heap_node_point_const_iterator_** (node_pointer p_nd)
- left_child_next_sibling_heap_node_point_const_iterator_ ()
- left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other)
- bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const
- const_reference operator∗ () const
- const_pointer operator-> () const
- bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Types

- typedef _Alloc::template
  rebind< Node >::other::pointer **node_pointer**

4.263.1 Detailed Description

template<typename Node, typename _Alloc>class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >

Const point-type iterator.

Definition at line 61 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2 Member Typedef Documentation

4.263.2.1 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::const_pointer

Iterator's const pointer type.

Definition at line 86 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2.2 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::const_reference

Iterator's const reference type.

Definition at line 98 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2.3 template<typename Node , typename _Alloc > typedef trivial_iterator_difference_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::difference_type

Difference type.

Definition at line 71 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2.4 template<typename Node , typename _Alloc > typedef trivial_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::iterator_category

Category.

Definition at line 68 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2.5 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::pointer

Iterator's pointer type.

Definition at line 80 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2.6 template<typename Node , typename _Alloc > typedef _Alloc::template rebind< value_type>::other::reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::reference

Iterator's reference type.

Definition at line 92 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.2.7 template<typename Node , typename _Alloc > typedef Node::value_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::value_type

Iterator's value type.

Definition at line 74 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.3 Constructor & Destructor Documentation

4.263.3.1 template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_ ( ) `[inline]`

Default constructor.

Definition at line 106 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.3.2 template<typename Node , typename _Alloc > __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & *other*) `[inline]`

Copy constructor.

Definition at line 111 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.4 Member Function Documentation

4.263.4.1 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & *other*) const `[inline]`

Compares content (negatively) to a different iterator object.

Definition at line 137 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.4.2 template<typename Node , typename _Alloc > const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator* ( ) const [inline]

Access.

Definition at line 124 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.4.3 template<typename Node , typename _Alloc > const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator-> ( ) const [inline]

Access.

Definition at line 116 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

4.263.4.4 template<typename Node , typename _Alloc > bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > & *other*) const [inline]

Compares content to a different iterator object.

Definition at line 132 of file left_child_next_sibling_heap_/point_const_iterator.hpp.

The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_/point_const_iterator.hpp

4.264 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference

Public Types

- typedef Size_Type **size_type**

Friends

- class **lu_counter_policy_base**< **size_type** >

4.264.1 Detailed Description

template<typename Size_Type = std::size_t>class __gnu_pbds::detail::lu_counter_metadata< Size_Type >

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

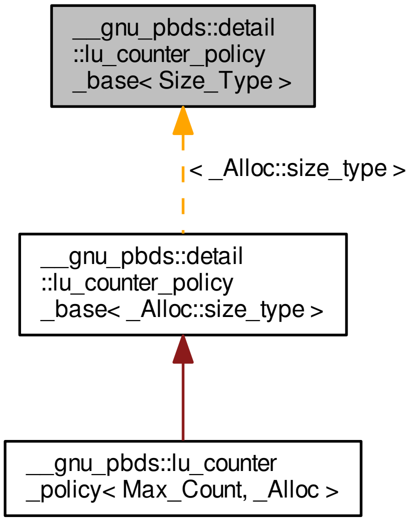
Definition at line 51 of file lu_counter_metadata.hpp.

The documentation for this class was generated from the following file:

- lu_counter_metadata.hpp

4.265 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::lu_counter_policy_base< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- lu_counter_metadata< size_type > **operator()** (size_type max_size) const
- template<typename Metadata_Reference >
  bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const

4.265.1 Detailed Description

template<typename Size_Type>class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >

Base class for list-update counter policy.

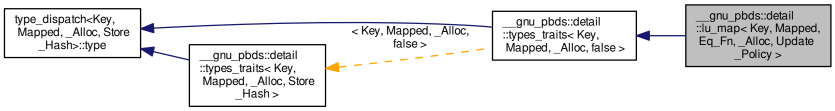
Definition at line 67 of file lu_counter_metadata.hpp.

The documentation for this class was generated from the following file:

- lu_counter_metadata.hpp

4.266 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >:



Public Types

- typedef _Alloc **allocator_type**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef Update_Policy::metadata_type **update_metadata**
- typedef Update_Policy **update_policy**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **lu_map** (const lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)
- template<typename It >
  lu_map (It, It)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred >
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- std::pair< point_iterator, bool > **insert** (const_reference)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It >
  void **copy_from_range** (It, It)

Friends

- class **const_iterator_**
- class **iterator_**

4.266.1 Detailed Description

template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >

list-based (with updates) associative container. Skip to the lu, my darling.

Definition at line 91 of file lu_map_.hpp.

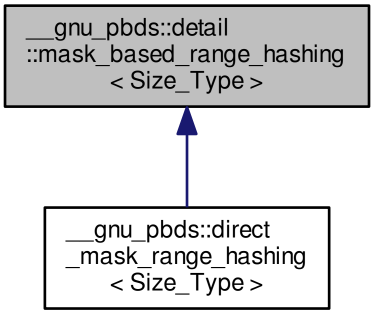
The documentation for this class was generated from the following files:

- lu_map_.hpp
- list_update_map_/constructor_destructor_fn_imps.hpp
- list_update_map_/info_fn_imps.hpp

- list_update_map_/iterators_fn_imps.hpp
- list_update_map_/erase_fn_imps.hpp
- list_update_map_/find_fn_imps.hpp
- list_update_map_/insert_fn_imps.hpp

4.267 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::mask_based_range_hashing< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

4.267.1 Detailed Description

template<typename Size_Type>class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >

Range hashing policy.

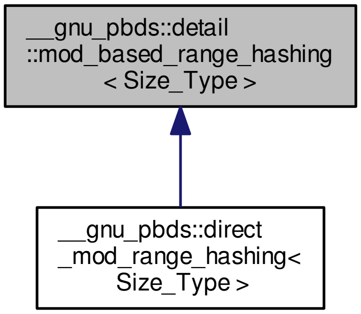
Definition at line 50 of file mask_based_range_hashing.hpp.

The documentation for this class was generated from the following file:

- mask_based_range_hashing.hpp

4.268 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::mod_based_range_hashing< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type s)
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

4.268.1 Detailed Description

template<typename Size_Type>class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >

Mod based range hashing.

Definition at line 50 of file mod_based_range_hashing.hpp.

The documentation for this class was generated from the following file:

- mod_based_range_hashing.hpp

4.269 __gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference

Public Types

- typedef integral_constant< int, __simple > **indicator**

Static Public Attributes

- static const bool **__simple**

4.269.1 Detailed Description

template<typename Key, typename Mapped>struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >

Primary template.

Definition at line 61 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.270 __gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference

Public Types

- typedef integral_constant< int,
  is_simple< Key >::value > **indicator**

4.270.1 Detailed Description

template<typename Key>struct __gnu_pbds::detail::no_throw_copies< Key, null_type >

Specialization.

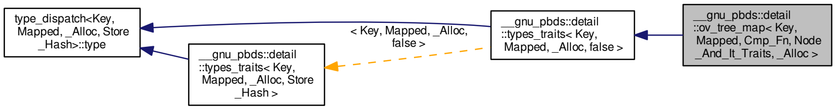
Definition at line 70 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.271 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >:



Classes

- class cond_dtor

  Conditional destructor.

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef ov_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef traits_type::node_update **node_update**
- typedef const_pointer **point_const_iterator**
- typedef pointer **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **ov_tree_map** (const Cmp_Fn &)
- **ov_tree_map** (const Cmp_Fn &, const node_update &)
- **ov_tree_map** (const ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()

- template<typename It >
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- iterator **erase** (iterator it)
- template<typename Pred >
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- std::pair< point_iterator, bool > **insert** (const_reference r_value)
- void **join** (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference r_key)
- point_const_iterator **lower_bound** (key_const_reference r_key) const
- size_type **max_size** () const
- node_const_iterator node_begin () const
- node_iterator node_begin ()
- node_const_iterator node_end () const
- node_iterator node_end ()
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **split** (key_const_reference, ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference r_key)
- point_const_iterator **upper_bound** (key_const_reference r_key) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

4.271.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Ordered-vector tree associative-container.

Definition at line 106 of file ov_tree_map_.hpp.

4.271.2 Member Function Documentation

4.271.2.1 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const `[inline]`

Returns a const node_iterator corresponding to the node at the root of the tree.

Definition at line 45 of file ov_tree_map_.hpp.

4.271.2.2 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) `[inline]`

Returns a node_iterator corresponding to the node at the root of the tree.

Definition at line 57 of file ov_tree_map_.hpp.

4.271.2.3 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const `[inline]`

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 51 of file ov_tree_map_.hpp.

4.271.2.4 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) `[inline]`

Returns a node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 63 of file ov_tree_map_.hpp.

The documentation for this class was generated from the following files:

- ov_tree_map_.hpp
- ov_tree_map_/constructors_destructor_fn_imps.hpp
- ov_tree_map_/iterators_fn_imps.hpp
- ov_tree_map_/erase_fn_imps.hpp
- ov_tree_map_/insert_fn_imps.hpp
- ov_tree_map_/info_fn_imps.hpp
- ov_tree_map_/split_join_fn_imps.hpp
- bin_search_tree_/policy_access_fn_imps.hpp

4.272 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference

Public Member Functions

- **cond_dtor** (value_vector a_vec, iterator &r_last_it, Size_Type total_size)
- void **set_no_action** ()

Protected Attributes

- value_vector **m_a_vec**
- const Size_Type **m_max_size**
- bool **m_no_action**
- iterator & **m_r_last_it**

4.272.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>template<typename Size_Type>class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >

Conditional destructor.

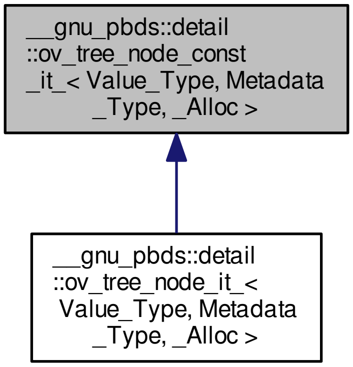
Definition at line 182 of file ov_tree_map_.hpp.

The documentation for this class was generated from the following file:

- ov_tree_map_.hpp

4.273 __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >:



Public Types

- typedef _Alloc::template rebind< typename remove_const < Value_Type >::type > ::other::const_pointer **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef _Alloc::template rebind< metadata_type > ::other::const_reference **metadata_const_reference**
- typedef Metadata_Type **metadata_type**

- typedef _Alloc::template rebind< typename remove_const < Value_Type >::type > ::other::const_pointer **reference**
- typedef _Alloc::template rebind< Value_Type > ::other::const_pointer **value_type**

Public Member Functions

- **ov_tree_node_const_it_** (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- this_type get_l_child () const
- metadata_const_reference **get_metadata** () const
- this_type get_r_child () const
- bool **operator!=** (const this_type &other) const
- const_reference **operator∗** () const
- bool **operator==** (const this_type &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Protected Types

- typedef _Alloc::template rebind< Metadata_Type > ::other::const_pointer **const_metadata_pointer**
- typedef _Alloc::template rebind< Value_Type > ::other::const_pointer **const_pointer**
- typedef _Alloc::template rebind< Value_Type > ::other::pointer **pointer**
- typedef ov_tree_node_const_it_ < Value_Type, Metadata_Type, _Alloc > **this_type**

Static Protected Member Functions

- template<typename Ptr >
  static Ptr **mid_pointer** (Ptr p_begin, Ptr p_end)

4.273.1 Detailed Description

template<typename Value_Type, typename Metadata_Type, typename _Alloc>class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >

Const node reference.

Definition at line 57 of file ov_tree_map_/node_iterators.hpp.

4.273.2 Member Function Documentation

4.273.2.1 template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_l_child ( ) const [inline]

Returns the node iterator associated with the left node.

Definition at line 142 of file ov_tree_map_/node_iterators.hpp.

4.273.2.2 template<typename Value_Type , typename Metadata_Type , typename _Alloc > this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_r_child ( ) const [inline]

Returns the node iterator associated with the right node.

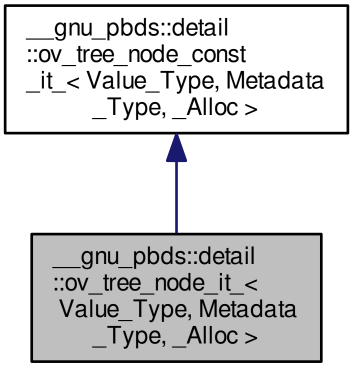
Definition at line 158 of file ov_tree_map_/node_iterators.hpp.

The documentation for this class was generated from the following file:

- ov_tree_map_/node_iterators.hpp

4.274 __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >:



Public Types

- typedef _Alloc::template rebind< typename remove_const < Value_Type >::type > ::other::pointer **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef _Alloc::template rebind< metadata_type > ::other::const_reference **metadata_const_reference**
- typedef Metadata_Type **metadata_type**
- typedef _Alloc::template rebind< typename remove_const < Value_Type >::type > ::other::pointer **reference**
- typedef _Alloc::template rebind< Value_Type > ::other::pointer **value_type**

Public Member Functions

- **ov_tree_node_it_** (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- ov_tree_node_it_ get_l_child () const
- metadata_const_reference **get_metadata** () const
- ov_tree_node_it_ get_r_child () const
- bool **operator!=** (const this_type &other) const
- reference operator* () const
- bool **operator==** (const this_type &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Static Protected Member Functions

- template<typename Ptr >
  static Ptr **mid_pointer** (Ptr p_begin, Ptr p_end)

4.274.1 Detailed Description

template<typename Value_Type, typename Metadata_Type, typename _Alloc>class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >

Node reference.

Definition at line 204 of file ov_tree_map_/node_iterators.hpp.

4.274.2 Member Function Documentation

4.274.2.1 template<typename Value_Type , typename Metadata_Type , typename _Alloc > ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get_l_child ( ) const [inline]

Returns the node reference associated with the left node.

Definition at line 252 of file ov_tree_map_/node_iterators.hpp.

4.274.2.2 template<typename Value_Type , typename Metadata_Type , typename _Alloc > ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get_r_child ( ) const [inline]

Returns the node reference associated with the right node.

Definition at line 268 of file ov_tree_map_/node_iterators.hpp.

4.274.2.3 template<typename Value_Type , typename Metadata_Type , typename _Alloc > reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* ( ) const [inline]

Access.

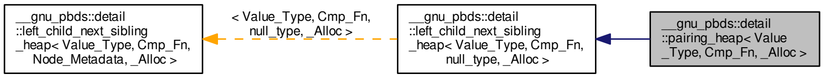
Definition at line 247 of file ov_tree_map_/node_iterators.hpp.

The documentation for this class was generated from the following file:

- ov_tree_map_/node_iterators.hpp

4.275 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_a::const_pointer **const_pointer**
- typedef __rebind_a::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_a::pointer **pointer**

- typedef __rebind_a::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **pairing_heap** (const Cmp_Fn &)
- **pairing_heap** (const pairing_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (pairing_heap &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
  void **split** (Pred, pairing_heap &)
- void **swap** (pairing_heap &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, null_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template
  rebind
  < left_child_next_sibling_heap_node_
  < Value_Type, null_type,
  _Alloc > >::other **node_allocator**
- typedef
  node_allocator::const_pointer **node_const_pointer**
- typedef null_type **node_metadata**
- typedef std::pair
  < node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
  void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.275.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >

Pairing heap.

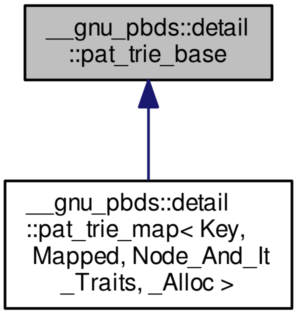
Definition at line 77 of file pairing_heap_.hpp.

The documentation for this class was generated from the following files:

- pairing_heap_.hpp
- pairing_heap_/constructors_destructor_fn_imps.hpp
- pairing_heap_/find_fn_imps.hpp
- pairing_heap_/insert_fn_imps.hpp
- pairing_heap_/erase_fn_imps.hpp
- pairing_heap_/split_join_fn_imps.hpp

4.276 __gnu_pbds::detail::pat_trie_base Struct Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base:



Classes

- class _CIter

  Const iterator.
- struct _Head

  Head node for PATRICIA tree.
- struct _Inode

  Internal node type, PATRICIA tree.
- class _Iter

  Iterator.
- struct _Leaf

  Leaf node for PATRICIA tree.
- struct _Metadata

  Metadata base primary template.
- struct _Metadata< null_type, _Alloc >

  Specialization for null metadata.
- struct _Node_base

  Node base.
- class _Node_citer

  Node const iterator.
- class _Node_iter

  Node iterator.

Public Types

- enum node_type { **i_node**, **leaf_node**, **head_node** }

4.276.1 Detailed Description

Base type for PATRICIA trees.

Definition at line 51 of file pat_trie_base.hpp.

4.276.2 Member Enumeration Documentation

4.276.2.1 enum __gnu_pbds::detail::pat_trie_base::node_type

Three types of nodes.

i_node is used by _Inode, leaf_node by _Leaf, and head_node by _Head.

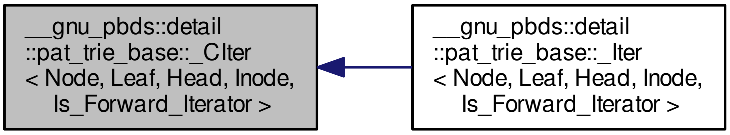
Definition at line 58 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.277 __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >:



Public Types

- typedef _Alloc::template rebind< Head > **__rebind_h**
- typedef _Alloc::template rebind< Inode > **__rebind_in**
- typedef _Alloc::template rebind< Leaf > **__rebind_l**
- typedef _Alloc::template rebind< Node > **__rebind_n**
- typedef allocator_type **_Alloc**
- typedef Node::allocator_type **allocator_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**

- typedef
  allocator_type::difference_type **difference_type**
- typedef __rebind_h::other::pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef __rebind_in::other::pointer **inode_pointer**
- typedef
  std::bidirectional_iterator_tag **iterator_category**
- typedef
  __rebind_l::other::const_pointer **leaf_const_pointer**
- typedef __rebind_l::other::pointer **leaf_pointer**
- typedef __rebind_n::other::pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef Node::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_CIter** (node_pointer p_nd=0)
- **_CIter** (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator!=** (const _CIter &other) const
- bool **operator!=** (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const
- const_reference **operator*** () const
- _CIter & **operator++** ()
- _CIter **operator++** (int)
- _CIter & **operator--** ()
- _CIter **operator--** (int)
- const_pointer **operator->** () const
- _CIter & **operator=** (const _CIter &other)
- _CIter & **operator=** (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator==** (const _CIter &other) const
- bool **operator==** (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

4.277.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

Const iterator.

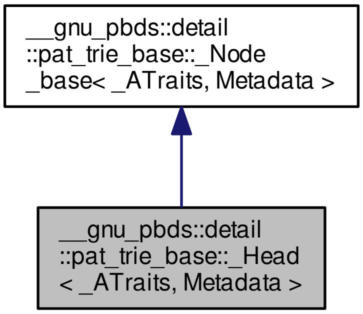
Definition at line 487 of file pat_trie_base.hpp.

The documentation for this class was generated from the following file:

- pat_trie_base.hpp

4.278 __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >:



Public Types

- typedef _Alloc::template rebind< _ATraits > **__rebind_at**
- typedef _Alloc::template rebind< _Node_base > **__rebind_n**
- typedef _ATraits::const_iterator **a_const_iterator**
- typedef __rebind_at::other::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef base_type::node_pointer **node_pointer**
- typedef base_type::type_traits **type_traits**

Public Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_min**
- node_pointer **m_p_parent**
- const node_type **m_type**

4.278.1 Detailed Description

template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >

Head node for PATRICIA tree.

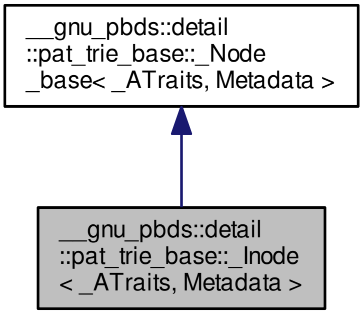
Definition at line 131 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.279 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >:



Classes

- struct const_iterator

    Constant child iterator.
- struct iterator

    Child iterator.

Public Types

- enum { **arr_size** }

- typedef _Alloc::template rebind< _ATraits > **__rebind_at**
- typedef _Alloc::template rebind< node_pointer >::other **__rebind_np**
- typedef base_type::allocator_type **_Alloc**
- typedef base_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef __rebind_np::pointer **node_pointer_pointer**
- typedef __rebind_np::reference **node_pointer_reference**
- typedef _Alloc::size_type **size_type**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Inode** (size_type, const a_const_iterator)
- node_pointer **add_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- const_iterator **begin** () const
- iterator **begin** ()
- const_iterator **end** () const
- iterator **end** ()
- iterator **get_child_it** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_const_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer) const
- size_type **get_e_ind** () const
- node_const_pointer **get_join_child** (node_const_pointer, a_const_pointer) const
- node_pointer **get_join_child** (node_pointer, a_const_pointer)
- node_pointer **get_lower_bound_child_node** (a_const_iterator, a_const_iterator, size_type, a_const_pointer)
- leaf_pointer **leftmost_descendant** ()
- leaf_const_pointer **leftmost_descendant** () const
- a_const_iterator **pref_b_it** () const
- a_const_iterator **pref_e_it** () const
- void **remove_child** (node_pointer)
- void **remove_child** (iterator)
- void **replace_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- leaf_pointer **rightmost_descendant** ()
- leaf_const_pointer **rightmost_descendant** () const
- bool **should_be_mine** (a_const_iterator, a_const_iterator, size_type, a_const_pointer) const
- void **update_prefixes** (a_const_pointer)

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

4.279.1 Detailed Description

template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >

Internal node type, PATRICIA tree.

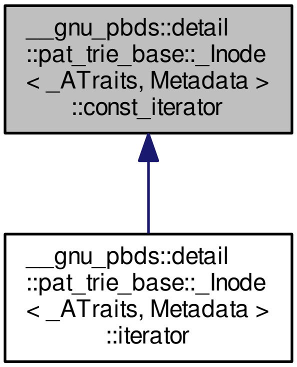
Definition at line 211 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.280 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator Struct Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator:



Public Types

- typedef _Alloc::difference_type **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **const_iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const const_iterator &other) const
- node_const_pointer **operator*** () const
- const_iterator & **operator++** ()
- const_iterator **operator++** (int)
- const node_pointer_pointer **operator->** () const
- bool **operator==** (const const_iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

4.280.1 Detailed Description

template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator

Constant child iterator.

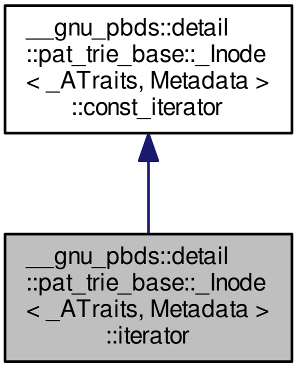
Definition at line 255 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.281 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator:



Public Types

- typedef _Alloc::difference_type **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const const_iterator &other) const
- bool **operator!=** (const iterator &other) const
- node_const_pointer **operator*** () const
- node_pointer **operator*** ()
- iterator & **operator++** ()
- iterator **operator++** (int)
- const node_pointer_pointer **operator->** () const
- node_pointer_pointer **operator->** ()
- bool **operator==** (const const_iterator &other) const
- bool **operator==** (const iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

4.281.1 Detailed Description

template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator

Child iterator.

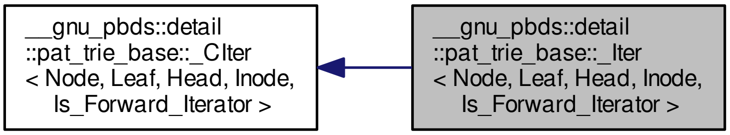
Definition at line 320 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.282 __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >:



Public Types

- typedef _Alloc::template rebind< Head > **__rebind_h**
- typedef _Alloc::template rebind< Inode > **__rebind_in**
- typedef _Alloc::template rebind< Leaf > **__rebind_l**
- typedef _Alloc::template rebind< Node > **__rebind_n**
- typedef allocator_type **_Alloc**
- typedef base_type::allocator_type **allocator_type**
- typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > **base_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::head_pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef base_type::inode_pointer **inode_pointer**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef base_type::leaf_const_pointer **leaf_const_pointer**
- typedef base_type::leaf_pointer **leaf_pointer**
- typedef base_type::node_pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Iter** (node_pointer p_nd=0)
- **_Iter** (const _Iter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator!=** (const _CIter &other) const
- bool **operator!=** (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const
- reference **operator∗** () const
- _Iter & **operator++** ()
- _Iter **operator++** (int)
- _Iter & **operator--** ()
- _Iter **operator--** (int)
- pointer **operator->** () const
- _Iter & **operator=** (const _Iter &other)
- _Iter & **operator=** (const _Iter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other)
- bool **operator==** (const _CIter &other) const
- bool **operator==** (const _CIter< Node, Leaf, Head, Inode,!Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

4.282.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

Iterator.

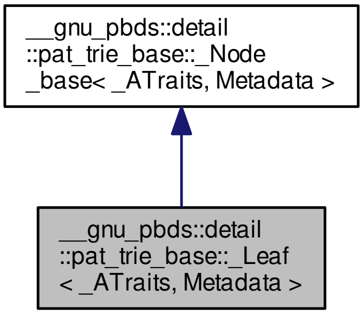
Definition at line 713 of file pat_trie_base.hpp.

The documentation for this class was generated from the following file:

- pat_trie_base.hpp

4.283 __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >:



Public Types

- typedef _Alloc::template rebind< _ATraits > **__rebind_at**
- typedef _Alloc::template rebind< _Node_base > **__rebind_n**
- typedef _ATraits::const_iterator **a_const_iterator**
- typedef __rebind_at::other::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef type_traits::const_reference **const_reference**
- typedef __rebind_n::other::pointer **node_pointer**
- typedef type_traits::reference **reference**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Leaf** (const_reference other)
- reference **value** ()
- const_reference **value** () const

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

4.283.1 Detailed Description

template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >

Leaf node for PATRICIA tree.

Definition at line 162 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.284 __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference

Public Types

- typedef _Alloc::template rebind< Metadata > **__rebind_m**
- typedef _Alloc **allocator_type**
- typedef __rebind_m::other::const_reference **const_reference**
- typedef Metadata **metadata_type**

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

4.284.1 Detailed Description

template<typename Metadata, typename _Alloc>struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >

Metadata base primary template.

Definition at line 67 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.285 __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference

Public Types

- typedef _Alloc **allocator_type**
- typedef null_type **metadata_type**

4.285.1 Detailed Description

template<typename _Alloc>struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >

Specialization for null metadata.

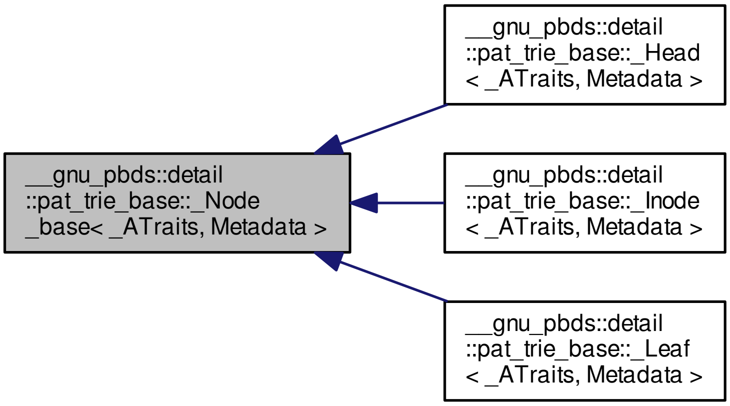
Definition at line 83 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.286 __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >:



Public Types

- typedef _Alloc::template rebind< _ATraits > **__rebind_at**
- typedef _Alloc::template rebind< _Node_base > **__rebind_n**
- typedef _ATraits::const_iterator **a_const_iterator**
- typedef __rebind_at::other::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef __rebind_n::other::pointer **node_pointer**
- typedef _ATraits::type_traits **type_traits**

Public Member Functions

- **_Node_base** (node_type type)

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

4.286.1 Detailed Description

template<typename _ATraits, typename Metadata>struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >

Node base.

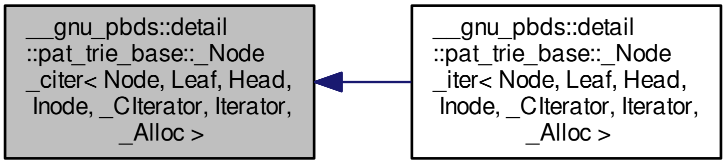
Definition at line 92 of file pat_trie_base.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

4.287 __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >:



Public Types

- typedef _Alloc::template rebind< metadata_type > __rebind_m
- typedef __rebind_m::other **__rebind_ma**
- typedef value_type **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef __rebind_ma::const_reference **metadata_const_reference**
- typedef Node::metadata_type metadata_type
- typedef value_type **reference**
- typedef _Alloc::size_type **size_type**
- typedef _CIterator **value_type**

Public Member Functions

- **_Node_citer** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- _Node_citer get_child (size_type i) const

- metadata_const_reference get_metadata () const
- size_type num_children () const
- bool operator!= (const _Node_citer &other) const
- const_reference operator* () const
- bool operator== (const _Node_citer &other) const
- std::pair< a_const_iterator,
  a_const_iterator > valid_prefix () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef _Alloc::template
  rebind< Inode > **__rebind_in**
- typedef _Alloc::template
  rebind< Leaf > **__rebind_l**
- typedef _Alloc::template
  rebind< Node > **__rebind_n**
- typedef Node::a_const_iterator **a_const_iterator**
- typedef Node::a_const_pointer **a_const_pointer**
- typedef
  __rebind_in::other::const_pointer **inode_const_pointer**
- typedef __rebind_in::other::pointer **inode_pointer**
- typedef
  __rebind_l::other::const_pointer **leaf_const_pointer**
- typedef __rebind_l::other::pointer **leaf_pointer**
- typedef __rebind_n::other::pointer **node_pointer**

4.287.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

Node const iterator.

Definition at line 814 of file pat_trie_base.hpp.

4.287.2 Member Typedef Documentation

4.287.2.1 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<metadata_type> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::__rebind_m

Const metadata reference type.

Definition at line 869 of file pat_trie_base.hpp.

4.287.2.2 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_type

Metadata type.

Definition at line 866 of file pat_trie_base.hpp.

4.287.3 Member Function Documentation

4.287.3.1 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > _Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child (size_type *i*) const [inline]

Returns a __const node __iterator to the corresponding node's i-th child.

Definition at line 911 of file pat_trie_base.hpp.

References std::advance().

4.287.3.2 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata ( ) const [inline]

Metadata access.

Definition at line 894 of file pat_trie_base.hpp.

4.287.3.3 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children ( ) const [inline]

Returns the number of children in the corresponding node.

Definition at line 899 of file pat_trie_base.hpp.

References std::distance().

Referenced by __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*(), and __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*().

4.287.3.4 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator!= (const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & *other*) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 927 of file pat_trie_base.hpp.

4.287.3.5 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator* ( ) const [inline]

Const access; returns the __const iterator* associated with the current leaf.

Definition at line 886 of file pat_trie_base.hpp.

References __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >-

::num_children().

4.287.3.6 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator== (const **_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & *other*) const [inline]

Compares content to a different iterator object.

Definition at line 922 of file pat_trie_base.hpp.

4.287.3.7 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > **std::pair**<a_const_iterator, a_const_iterator> **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix ( ) const [inline]

Subtree valid prefix.

Definition at line 880 of file pat_trie_base.hpp.

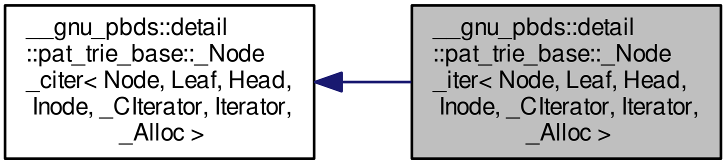
References std::make_pair().

The documentation for this class was generated from the following file:

- pat_trie_base.hpp

4.288 __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >:



Public Types

- typedef _Alloc::template rebind< metadata_type > __rebind_m
- typedef __rebind_m::other **__rebind_ma**
- typedef value_type **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**

- typedef __rebind_ma::const_reference **metadata_const_reference**
- typedef Node::metadata_type metadata_type
- typedef value_type **reference**
- typedef base_type::size_type **size_type**
- typedef Iterator **value_type**

Public Member Functions

- **_Node_iter** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- _Node_iter get_child (size_type i) const
- metadata_const_reference get_metadata () const
- size_type num_children () const
- bool operator!= (const _Node_citer &other) const
- reference operator* () const
- bool operator== (const _Node_citer &other) const
- std::pair< a_const_iterator, a_const_iterator > valid_prefix () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef _Alloc::template rebind< Inode > **__rebind_in**
- typedef _Alloc::template rebind< Leaf > **__rebind_l**
- typedef Node::a_const_iterator **a_const_iterator**
- typedef __rebind_in::other::const_pointer **inode_const_pointer**
- typedef __rebind_l::other::const_pointer **leaf_const_pointer**
- typedef __rebind_l::other::pointer **leaf_pointer**

4.288.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

Node iterator.

Definition at line 943 of file pat_trie_base.hpp.

4.288.2 Member Typedef Documentation

4.288.2.1 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef _Alloc::template rebind<**metadata_type**> **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::**__rebind_m** `[inherited]`

Const metadata reference type.

Definition at line 869 of file pat_trie_base.hpp.

4.288.2.2 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > typedef Node::metadata_type **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::**metadata_type** `[inherited]`

Metadata type.

Definition at line 866 of file pat_trie_base.hpp.

4.288.3 Member Function Documentation

4.288.3.1 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > **_Node_iter** **__gnu_pbds::detail::pat_trie_base::_Node_iter**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_child (size_type *i*) const `[inline]`

Returns a node __iterator to the corresponding node's i-th child.

Definition at line 976 of file pat_trie_base.hpp.

References std::advance(), and std::begin().

4.288.3.2 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > metadata_const_reference **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::get_metadata ( ) const `[inline]`, `[inherited]`

Metadata access.

Definition at line 894 of file pat_trie_base.hpp.

4.288.3.3 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > size_type **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children ( ) const `[inline]`, `[inherited]`

Returns the number of children in the corresponding node.

Definition at line 899 of file pat_trie_base.hpp.

References std::distance().

Referenced by __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*(), and __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator*().

4.288.3.4 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool **__gnu_pbds::detail::pat_trie_base::_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator!= (const **_Node_citer**< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & *other*) const `[inline]`, `[inherited]`

Compares content (negatively) to a different iterator object.

Definition at line 927 of file pat_trie_base.hpp.

4.288.3.5 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator∗ ( ) const [inline]

Access; returns the iterator∗ associated with the current leaf.

Definition at line 968 of file pat_trie_base.hpp.

References __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::num_children().

4.288.3.6 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::operator== (const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & *other*) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 922 of file pat_trie_base.hpp.

4.288.3.7 template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator , typename Iterator , typename _Alloc > std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix ( ) const [inline], [inherited]

Subtree valid prefix.

Definition at line 880 of file pat_trie_base.hpp.

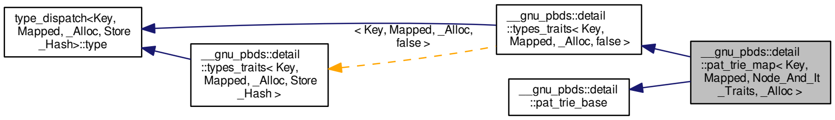
References std::make_pair().

The documentation for this class was generated from the following file:

- pat_trie_base.hpp

4.289 __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >:



Public Types

- typedef traits_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**

- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef traits_type::const_reverse_iterator **const_reverse_iterator**
- typedef pat_trie_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- enum node_type { **i_node**, **leaf_node**, **head_node** }
- typedef traits_type::node_update **node_update**
- typedef traits_type::const_iterator **point_const_iterator**
- typedef traits_type::iterator **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef traits_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **pat_trie_map** (const access_traits &)
- **pat_trie_map** (const pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)

- const_iterator **erase** (const_iterator)
- iterator **erase** (iterator)
- const_reverse_iterator **erase** (const_reverse_iterator)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- access_traits & **get_access_traits** ()
- const access_traits & **get_access_traits** () const
- node_update & **get_node_update** ()
- const node_update & **get_node_update** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator node_begin () const
- node_iterator node_begin ()
- node_const_iterator node_end () const
- node_iterator node_end ()
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- void **swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It >
  void **copy_from_range** (It, It)
- node_pointer **recursive_copy_node** (node_const_pointer)
- void **value_swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)

4.289.1 Detailed Description

template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Definition at line 101 of file pat_trie_.hpp.

4.289.2 Member Enumeration Documentation

4.289.2.1 enum __gnu_pbds::detail::pat_trie_base::node_type [inherited]

Three types of nodes.

i_node is used by _Inode, leaf_node by _Leaf, and head_node by _Head.

Definition at line 58 of file pat_trie_base.hpp.

4.289.3 Member Function Documentation

4.289.3.1 template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ( ) const [inline]

Returns a const node_iterator corresponding to the node at the root of the tree.

Definition at line 101 of file pat_trie_.hpp.

4.289.3.2 template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ( ) [inline]

Returns a node_iterator corresponding to the node at the root of the tree.

Definition at line 107 of file pat_trie_.hpp.

4.289.3.3 template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ( ) const [inline]

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 113 of file pat_trie_.hpp.

4.289.3.4 template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc > pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ( ) [inline]

Returns a node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 119 of file pat_trie_.hpp.

The documentation for this class was generated from the following files:

- pat_trie_.hpp
- pat_trie_/constructors_destructor_fn_imps.hpp
- pat_trie_/iterators_fn_imps.hpp
- insert_join_fn_imps.hpp
- pat_trie_/erase_fn_imps.hpp
- pat_trie_/find_fn_imps.hpp
- pat_trie_/info_fn_imps.hpp
- pat_trie_/policy_access_fn_imps.hpp
- split_fn_imps.hpp
- update_fn_imps.hpp

4.290 __gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference

4.290.1 Detailed Description

template<typename _Alloc>class __gnu_pbds::detail::probe_fn_base< _Alloc >

Probe functor base.

Definition at line 52 of file probe_fn_base.hpp.

The documentation for this class was generated from the following file:

- probe_fn_base.hpp

4.291 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference

Inherits Hash_Fn, and Comb_Hash_Fn.

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef Hash_Fn **hash_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference) const
- void **swap** (ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > &)

4.291.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

Definition at line 71 of file ranged_hash_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

4.292 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference

Inherits Hash_Fn, and Comb_Hash_Fn.

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- comp_hash **operator()** (key_const_reference) const
- comp_hash **operator()** (key_const_reference, size_type) const
- void **swap** (ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > &)

4.292.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

Definition at line 153 of file ranged_hash_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

4.293 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference

Inherits Comb_Hash_Fn.

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const null_type &, const Comb_Hash_Fn &)
- void **swap** (ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > &)

4.293.1 Detailed Description

template<typename Key, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >

Specialization 3 The client does not supply a hash function (by specifying null_type as the Hash_Fn parameter), and requests that hash values not be stored.

Definition at line 255 of file ranged_hash_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

4.294 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference

Inherits Comb_Hash_Fn.

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const null_type &, const Comb_Hash_Fn &)
- void **swap** (ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > &)

4.294.1 Detailed Description

template<typename Key, typename _Alloc, typename Comb_Hash_Fn>class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >

Specialization 4 The client does not supply a hash function (by specifying null_type as the Hash_Fn parameter), and requests that hash values be stored.

Definition at line 312 of file ranged_hash_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

4.295 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference

Inherits Hash_Fn, Comb_Probe_Fn, and Probe_Fn.

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef Hash_Fn **hash_fn_base**
- typedef _Alloc::template
  rebind< Key >::other **key_allocator**
- typedef
  key_allocator::const_reference **key_const_reference**
- typedef Probe_Fn **probe_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type)
- **ranged_probe_fn** (size_type, const Hash_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference) const
- size_type **operator()** (key_const_reference, size_type, size_type) const
- void **swap** (ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > &)

4.295.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >

Specialization 1 The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 71 of file ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

4.296 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference

Inherits Hash_Fn, Comb_Probe_Fn, and Probe_Fn.

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef Probe_Fn **probe_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type)
- **ranged_probe_fn** (size_type, const Hash_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- void **notify_resized** (size_type)
- comp_hash **operator()** (key_const_reference) const
- size_type **operator()** (key_const_reference, size_type, size_type) const
- size_type **operator()** (key_const_reference, size_type) const
- void **swap** (ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > &)

4.296.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 176 of file ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

4.297 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference

Inherits Comb_Probe_Fn.

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef _Alloc::template rebind< Key >::other **key_allocator**
- typedef key_allocator::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type size)
- **ranged_probe_fn** (size_type, const Comb_Probe_Fn &r_comb_probe_fn)
- **ranged_probe_fn** (size_type, const null_type &, const Comb_Probe_Fn &r_comb_probe_fn, const null_type &)
- void **swap** (ranged_probe_fn &other)

4.297.1 Detailed Description

template<typename Key, typename _Alloc, typename Comb_Probe_Fn>class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

Definition at line 296 of file ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

4.298 __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

Inherits __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >.

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef base_type::const_reverse_iterator **const_reverse_iterator**
- typedef rb_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::key_const_pointer **key_const_pointer**

- typedef
  base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef
  base_type::mapped_const_pointer **mapped_const_pointer**
- typedef
  base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef
  traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int,
  Store_Hash > **store_extra**
- typedef base_type::value_type **value_type**

Public Member Functions

- **rb_tree_map** (const Cmp_Fn &)
- **rb_tree_map** (const Cmp_Fn &, const node_update &)
- **rb_tree_map** (const rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- iterator **erase** (iterator)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator **node_begin** () const
- node_iterator **node_begin** ()
- node_const_iterator **node_end** () const
- node_iterator **node_end** ()
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template
  rebind< typename
  traits_type::node >::other **node_allocator**
- typedef
  traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef types_traits< Key,
  Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **apply_update** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_ >
  void **apply_update** (node_pointer, Node_Update_ ∗)
- std::pair< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- std::pair< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- template<typename Node_Update_ >
  void **update_to_top** (node_pointer, Node_Update_ ∗)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

4.298.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

Definition at line 84 of file rb_tree_.hpp.

4.298.2 Member Function Documentation

4.298.2.1 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const `[inline]`, `[inherited]`

Returns a const node_iterator corresponding to the node at the root of the tree.

Definition at line 109 of file bin_search_tree_.hpp.

4.298.2.2 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) `[inline]`, `[inherited]`

Returns a node_iterator corresponding to the node at the root of the tree.

Definition at line 117 of file bin_search_tree_.hpp.

4.298.2.3 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const `[inline]`, `[inherited]`

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 125 of file bin_search_tree_.hpp.

4.298.2.4 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) `[inline]`, `[inherited]`

Returns a node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 133 of file bin_search_tree_.hpp.

The documentation for this class was generated from the following files:

- rb_tree_.hpp
- rb_tree_map_/constructors_destructor_fn_imps.hpp
- rb_tree_map_/insert_fn_imps.hpp
- rb_tree_map_/erase_fn_imps.hpp
- rb_tree_map_/split_join_fn_imps.hpp
- rb_tree_map_/info_fn_imps.hpp

4.299 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

Public Types

- typedef _Alloc::template
  rebind< metadata_type >
  ::other::const_reference **metadata_const_reference**
- typedef _Alloc::template
  rebind< metadata_type >
  ::other::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef _Alloc::template
  rebind< rb_tree_node_
  < Value_Type, Metadata, _Alloc >
  >::other::pointer **node_pointer**
- typedef Value_Type **value_type**

Public Member Functions

- metadata_const_reference **get_metadata** () const
- metadata_reference **get_metadata** ()
- bool **special** () const

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_left**
- node_pointer **m_p_parent**
- node_pointer **m_p_right**
- bool **m_red**
- value_type **m_value**

4.299.1 Detailed Description

template<typename Value_Type, class Metadata, typename _Alloc>struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >

Node for Red-Black trees.

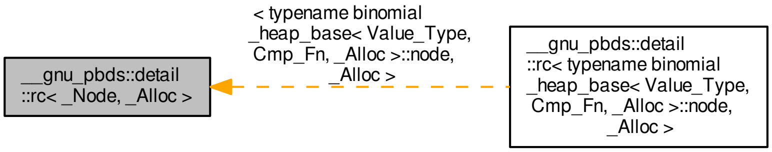
Definition at line 52 of file rb_tree_map_/node.hpp.

The documentation for this struct was generated from the following file:

- rb_tree_map_/node.hpp

4.300 __gnu_pbds::detail::rc< _Node, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::rc< _Node, _Alloc >:



Public Types

- typedef entry_const_pointer **const_iterator**
- typedef node_pointer **entry**

Public Member Functions

- **rc** (const rc &)
- const const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- const const_iterator **end** () const
- void **pop** ()
- void **push** (entry)
- size_type **size** () const
- void **swap** (rc &)
- node_pointer **top** () const

4.300.1 Detailed Description

template<typename _Node, typename _Alloc>class __gnu_pbds::detail::rc< _Node, _Alloc >

Redundant binary counter.

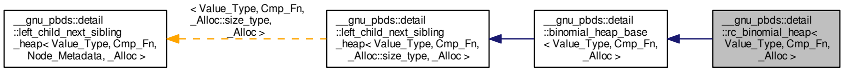
Definition at line 50 of file rc.hpp.

The documentation for this class was generated from the following file:

- rc.hpp

4.301 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **rc_binomial_heap** (const Cmp_Fn &)
- **rc_binomial_heap** (const rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
  void **split** (Pred, rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- template<typename Pred >
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template
  rebind
  < left_child_next_sibling_heap_node_
  < Value_Type,
  _Alloc::size_type, _Alloc >
  >::other **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair
  < node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
  void **copy_from_range** (It, It)
- void **find_max** ()

- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

4.301.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Redundant-counter binomial heap.

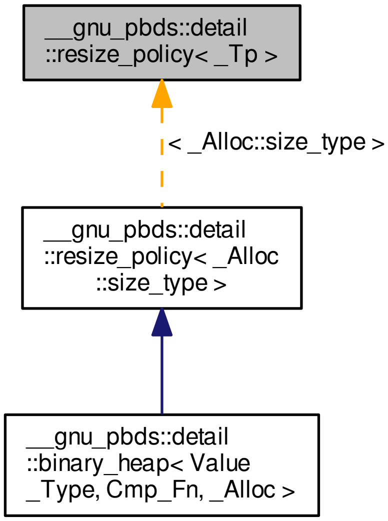
Definition at line 66 of file rc_binomial_heap_.hpp.

The documentation for this class was generated from the following files:

- rc_binomial_heap_.hpp
- rc_binomial_heap_/constructors_destructor_fn_imps.hpp
- rc_binomial_heap_/erase_fn_imps.hpp
- rc_binomial_heap_/insert_fn_imps.hpp
- rc_binomial_heap_/split_join_fn_imps.hpp

4.302 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::resize_policy< _Tp >:



Public Types

- typedef _Tp **size_type**

Public Member Functions

- **resize_policy** (const resize_policy &other)
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- bool **resize_needed_for_grow** (size_type) const
- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- void **swap** (resize_policy< _Tp > &)

Static Public Attributes

- static const _Tp **min_size**

4.302.1 Detailed Description

template<typename _Tp>class __gnu_pbds::detail::resize_policy< _Tp >

Resize policy for binary heap.

Definition at line 52 of file resize_policy.hpp.

The documentation for this class was generated from the following file:

- resize_policy.hpp

4.303 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

Inherits __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >.

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef base_type::const_reverse_iterator **const_reverse_iterator**
- typedef splay_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::key_const_pointer **key_const_pointer**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef base_type::mapped_const_pointer **mapped_const_pointer**
- typedef base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**

- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int,
  Store_Hash > **store_extra**
- typedef base_type::value_type **value_type**

Public Member Functions

- **splay_tree_map** (const Cmp_Fn &)
- **splay_tree_map** (const Cmp_Fn &, const node_update &)
- **splay_tree_map** (const splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It >
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- iterator **erase** (iterator it)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **initialize** ()
- std::pair< point_iterator, bool > **insert** (const_reference r_value)
- void **join** (splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_const_iterator **node_begin** () const
- node_iterator **node_begin** ()
- node_const_iterator **node_end** () const
- node_iterator **node_end** ()
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_allocator::value_type **node**
- typedef _Alloc::template rebind< typename traits_type::node >::other **node_allocator**
- typedef traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef types_traits< Key, Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **apply_update** (node_pointer, null_node_update_pointer)
- template< typename Node_Update_ >
  void **apply_update** (node_pointer, Node_Update_ ∗)
- std::pair< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- std::pair< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- template< typename Node_Update_ >
  void **update_to_top** (node_pointer, Node_Update_ ∗)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

4.303.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Splay tree.

Definition at line 107 of file splay_tree_.hpp.

4.303.2 Member Function Documentation

4.303.2.1 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) const `[inline]`, `[inherited]`

Returns a const node_iterator corresponding to the node at the root of the tree.

Definition at line 109 of file bin_search_tree_.hpp.

4.303.2.2 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ( ) `[inline]`, `[inherited]`

Returns a node_iterator corresponding to the node at the root of the tree.

Definition at line 117 of file bin_search_tree_.hpp.

4.303.2.3 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) const `[inline]`, `[inherited]`

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 125 of file bin_search_tree_.hpp.

4.303.2.4 template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename _Alloc > bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ( ) `[inline]`, `[inherited]`

Returns a node_iterator corresponding to a node just after a leaf of the tree.

Definition at line 133 of file bin_search_tree_.hpp.

The documentation for this class was generated from the following files:

- splay_tree_.hpp
- splay_tree_/constructors_destructor_fn_imps.hpp
- splay_tree_/insert_fn_imps.hpp
- splay_fn_imps.hpp
- splay_tree_/erase_fn_imps.hpp
- splay_tree_/find_fn_imps.hpp
- splay_tree_/split_join_fn_imps.hpp

4.304 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

Public Types

- typedef _Alloc::template
  rebind< metadata_type >
  ::other::const_reference **metadata_const_reference**
- typedef _Alloc::template
  rebind< metadata_type >
  ::other::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef _Alloc::template
  rebind< splay_tree_node_
  < Value_Type, Metadata, _Alloc >
  >::other::pointer **node_pointer**
- typedef Value_Type **value_type**

Public Member Functions

- metadata_const_reference **get_metadata** () const
- metadata_reference **get_metadata** ()
- bool **special** () const

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_left**
- node_pointer **m_p_parent**
- node_pointer **m_p_right**
- bool **m_special**
- value_type **m_value**

4.304.1 Detailed Description

template<typename Value_Type, class Metadata, typename _Alloc>struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >

Node for splay tree.

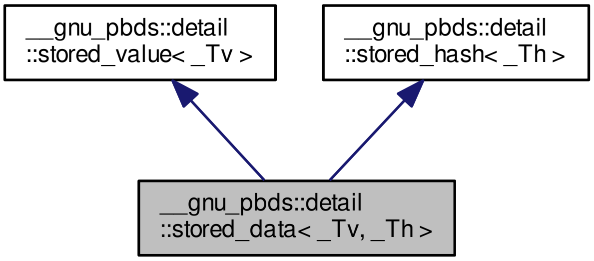
Definition at line 50 of file splay_tree_/node.hpp.

The documentation for this struct was generated from the following file:

- splay_tree_/node.hpp

4.305 __gnu_pbds::detail::stored_data< _Tv, _Th > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::stored_data< _Tv, _Th >:



Public Types

- typedef _Th **hash_type**
- typedef _Tv **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

4.305.1 Detailed Description

template<typename _Tv, typename _Th>struct __gnu_pbds::detail::stored_data< _Tv, _Th >

Primary template for representation of stored data. Two types of data can be stored: value and hash.

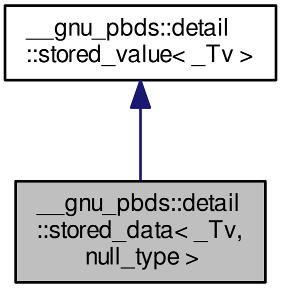
Definition at line 95 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.306 __gnu_pbds::detail::stored_data< _Tv, null_type > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::stored_data< _Tv, null_type >:



Public Types

- typedef _Tv **value_type**

Public Attributes

- value_type **m_value**

4.306.1 Detailed Description

template<typename _Tv>struct __gnu_pbds::detail::stored_data< _Tv, null_type >

Specialization for representation of stored data of just value type.

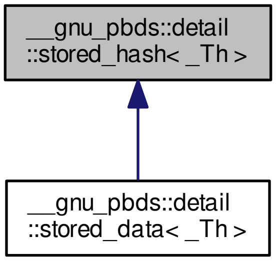
Definition at line 101 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.307 __gnu_pbds::detail::stored_hash< _Th > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::stored_hash< _Th >:



Public Types

- typedef _Th **hash_type**

Public Attributes

- hash_type **m_hash**

4.307.1 Detailed Description

template<typename _Th>struct __gnu_pbds::detail::stored_hash< _Th >

Stored hash.

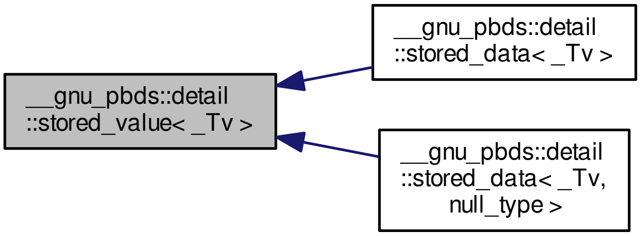
Definition at line 86 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.308 __gnu_pbds::detail::stored_value< _Tv > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::stored_value< _Tv >:



Public Types

- typedef _Tv **value_type**

Public Attributes

- value_type **m_value**

4.308.1 Detailed Description

template<typename _Tv>struct __gnu_pbds::detail::stored_value< _Tv >

Stored value.

Definition at line 78 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.309 __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference

Inherits _ATraits.

Public Types

- typedef _ATraits **base_type**
- typedef base_type::const_iterator **const_iterator**
- typedef
  type_traits::const_reference **const_reference**
- typedef
  type_traits::key_const_reference **key_const_reference**

- typedef Type_Traits **type_traits**

Public Member Functions

- **synth_access_traits** (const base_type &)
- bool **cmp_keys** (key_const_reference, key_const_reference) const
- bool **cmp_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=false) const
- bool **equal_keys** (key_const_reference, key_const_reference) const
- bool **equal_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=true) const

Static Public Member Functions

- static key_const_reference **extract_key** (const_reference)

4.309.1 Detailed Description

template<typename Type_Traits, bool Set, typename _ATraits>struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >

Synthetic element access traits.

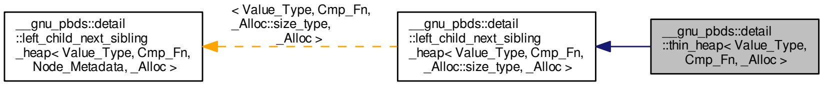
Definition at line 59 of file synth_access_traits.hpp.

The documentation for this struct was generated from the following file:

- synth_access_traits.hpp

4.310 __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_a::const_pointer **const_pointer**
- typedef __rebind_a::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**

- typedef
  base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_a::pointer **pointer**
- typedef __rebind_a::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred >
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred >
  void **split** (Pred, thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef _Alloc::template
  rebind
  < left_child_next_sibling_heap_node_
  < Value_Type,
  _Alloc::size_type, _Alloc >
  >::other **node_allocator**
- typedef
  base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef std::pair
  < node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **thin_heap** (const Cmp_Fn &)
- **thin_heap** (const thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It >
  void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

4.310.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Thin heap.

See Tarjan and Kaplan.

Definition at line 77 of file thin_heap_.hpp.

The documentation for this class was generated from the following files:

- thin_heap_.hpp
- thin_heap_/constructors_destructor_fn_imps.hpp
- thin_heap_/find_fn_imps.hpp
- thin_heap_/insert_fn_imps.hpp
- thin_heap_/erase_fn_imps.hpp
- thin_heap_/split_join_fn_imps.hpp

4.311 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference

Public Types

- typedef Node_Update::metadata_type **type**

4.311.1 Detailed Description

template<typename Node_Update>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >

Specialization, false.

Definition at line 62 of file tree_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

4.312 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference

Public Types

- typedef null_type **type**

4.312.1 Detailed Description

template<typename Node_Update>struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >

Specialization, true.

Definition at line 69 of file tree_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

4.313 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

Public Types

- typedef tree_metadata_helper
  < __node_u, null_update >
  ::type **type**

4.313.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Tree node metadata dispatch.

Definition at line 84 of file tree_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

4.314 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference

Public Types

- typedef
  tree_node_metadata_dispatch
  < Key, Mapped, Cmp_Fn,
  Node_Update, _Alloc >::type **metadata_type**
- typedef ov_tree_node_const_it_
  < value_type, metadata_type,
  _Alloc > node_const_iterator
- typedef ov_tree_node_it_
  < value_type, metadata_type,
  _Alloc > **node_iterator**
- typedef Node_Update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef
  __gnu_pbds::null_node_update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**

4.314.1 Detailed Description

template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Tree traits.

Definition at line 61 of file ov_tree_map_/traits.hpp.

4.314.2 Member Typedef Documentation

4.314.2.1 template<typename Key , typename Mapped , class Cmp_Fn , template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

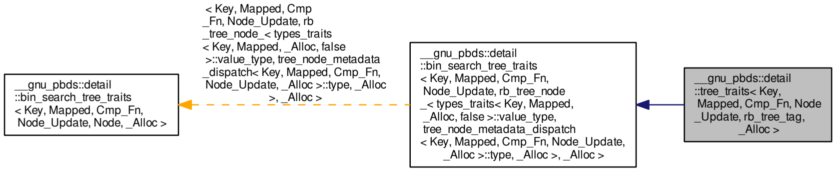
Definition at line 95 of file ov_tree_map_/traits.hpp.

The documentation for this struct was generated from the following file:

- ov_tree_map_/traits.hpp

4.315 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef rb_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_ < rb_tree_node_< types_traits < Key, Mapped, _Alloc, false > ::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef

bin_search_tree_node_it_
< rb_tree_node_< types_traits
< Key, Mapped, _Alloc, false >
::value_type,
tree_node_metadata_dispatch
< Key, Mapped, Cmp_Fn,
Node_Update, _Alloc >::type,
_Alloc >, point_const_iterator,
point_iterator, _Alloc > **node_iterator**

- typedef Node_Update
< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef
__gnu_pbds::null_node_update
< node_const_iterator,
node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**
- typedef
bin_search_tree_const_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_

< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > **reverse_iterator**

4.315.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Specialization.

Definition at line 61 of file rb_tree_map_/traits.hpp.

4.315.2 Member Typedef Documentation

4.315.2.1 typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , _Alloc >::node_const_iterator `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

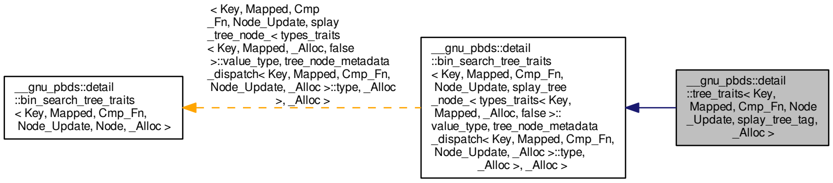
Definition at line 131 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- rb_tree_map_/traits.hpp

4.316 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef splay_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_ < splay_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef

bin_search_tree_node_it_ < splay_tree_node_ < types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**

- typedef Node_Update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**
- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_

< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > **reverse_iterator**

4.316.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Specialization.

Definition at line 61 of file splay_tree_/traits.hpp.

4.316.2 Member Typedef Documentation

4.316.2.1 typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , _Alloc >::node_const_iterator `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- splay_tree_/traits.hpp

4.317 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference

Public Types

- typedef
tree_node_metadata_dispatch
< Key, null_type, Cmp_Fn,
Node_Update, _Alloc >::type **metadata_type**
- typedef ov_tree_node_const_it_
< value_type, metadata_type,
_Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**

- typedef Node_Update
  < node_const_iterator,
  node_const_iterator, Cmp_Fn,
  _Alloc > **node_update**
- typedef
  __gnu_pbds::null_node_update
  < node_const_iterator,
  node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**

4.317.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Specialization.

Definition at line 132 of file ov_tree_map_/traits.hpp.

4.317.2 Member Typedef Documentation

4.317.2.1 template<typename Key , class Cmp_Fn , template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

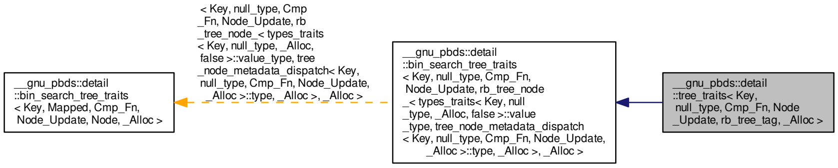
Definition at line 166 of file ov_tree_map_/traits.hpp.

The documentation for this struct was generated from the following file:

- ov_tree_map_/traits.hpp

4.318 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef rb_tree_node_ < types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_ < rb_tree_node_< types_traits < Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_ < rb_tree_node_< types_traits < Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**
- typedef

bin_search_tree_const_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_const_iterator**

- typedef bin_search_tree_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > **reverse_iterator**

4.318.1 Detailed Description

template<typename Key, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Specialization.

Definition at line 85 of file rb_tree_map_/traits.hpp.

4.318.2 Member Typedef Documentation

4.318.2.1 typedef **bin_search_tree_const_node_it_< rb_tree_node_< types_traits**< Key, **null_type**, _Alloc, false >::value_type, **tree_node_metadata_dispatch**< Key, **null_type**, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , **point_const_iterator**, **point_iterator**, _Alloc> **__gnu_pbds::detail::bin_search_tree_traits**< Key, **null_type** , Cmp_Fn, Node_Update, **rb_tree_node_< types_traits**< Key, **null_type**, _Alloc, false >::value_type, **tree_node_metadata_dispatch**< Key, **null_type**, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > , _Alloc >::**node_const_iterator** `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

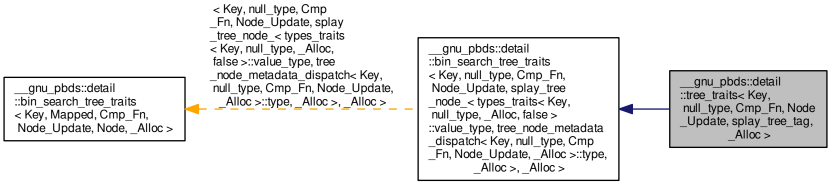
Definition at line 131 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- rb_tree_map_/traits.hpp

4.319 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef splay_tree_node_ < types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type,

_Alloc > **node**

- typedef bin_search_tree_const_node_it_ < splay_tree_node_ < types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_ < splay_tree_node_ < types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update < node_const_iterator, node_iterator, Cmp_Fn, _Alloc > ∗ **null_node_update_pointer**
- typedef bin_search_tree_const_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_ < typename _Alloc::template rebind< node >::other::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference,

true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_
< typename _Alloc::template
rebind< node >::other::pointer,
typename
type_traits::value_type,
typename type_traits::pointer,
typename
type_traits::const_pointer,
typename
type_traits::reference,
typename
type_traits::const_reference,
false, _Alloc > **reverse_iterator**

4.319.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Specialization.

Definition at line 81 of file splay_tree_/traits.hpp.

4.319.2 Member Typedef Documentation

4.319.2.1 typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator `[inherited]`

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 131 of file bin_search_tree_/traits.hpp.

The documentation for this struct was generated from the following file:

- splay_tree_/traits.hpp

4.320 __gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference

Public Types

- typedef Node_Update::metadata_type **type**

4.320.1 Detailed Description

template<typename Node_Update>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >

Specialization, false.

Definition at line 62 of file trie_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

4.321 __gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference

Public Types

- typedef null_type **type**

4.321.1 Detailed Description

template<typename Node_Update>struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >

Specialization, true.

Definition at line 69 of file trie_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

4.322 __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

Public Types

- typedef trie_metadata_helper
  < __node_u, null_update >
  ::type **type**

4.322.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Trie node metadata dispatch.

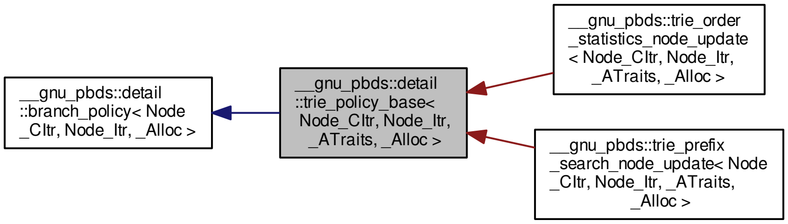
Definition at line 84 of file trie_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

4.323 __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef null_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef _Alloc::template rebind< rckey_type >::other **rebind_k**
- typedef _Alloc::template rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual const_iterator **end** () const =0
- virtual iterator **end** ()=0
- it_type **end_iterator** () const
- virtual const access_traits & **get_access_traits** () const =0
- virtual node_const_iterator **node_begin** () const =0
- virtual node_iterator **node_begin** ()=0
- virtual node_const_iterator **node_end** () const =0
- virtual node_iterator **node_end** ()=0

Static Protected Member Functions

- static size_type **common_prefix_len** (node_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static key_const_reference **extract_key** (const_reference r_val)
- static iterator **leftmost_it** (node_iterator)
- static bool **less** (e_const_iterator, e_const_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static iterator **rightmost_it** (node_iterator)

4.323.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >

Base class for trie policies.

Definition at line 53 of file trie_policy_base.hpp.

The documentation for this class was generated from the following file:

- trie_policy_base.hpp

4.324 __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference

Public Types

- typedef _ATraits **access_traits**
- typedef base_type::_CIter < node, leaf, head, inode, true > **const_iterator**
- typedef base_type::_CIter < node, leaf, head, inode, false > **const_reverse_iterator**
- typedef base_type::_Head < synth_access_traits, metadata > **head**
- typedef base_type::_Inode < synth_access_traits, metadata > **inode**
- typedef base_type::_Iter< node, leaf, head, inode, true > **iterator**

- typedef base_type::_Leaf < synth_access_traits, metadata > **leaf**
- typedef base_type::_Metadata < metadata_type, _Alloc > **metadata**
- typedef trie_node_metadata_dispatch < Key, Mapped, _ATraits, Node_Update, _Alloc >::type **metadata_type**
- typedef base_type::_Node_base < synth_access_traits, metadata > **node**
- typedef base_type::_Node_citer < node, leaf, head, inode, const_iterator, iterator, _Alloc > node_const_iterator
- typedef base_type::_Node_iter < node, leaf, head, inode, const_iterator, iterator, _Alloc > **node_iterator**
- typedef Node_Update < node_const_iterator, node_iterator, _ATraits, _Alloc > node_update
- typedef null_node_update < node_const_iterator, node_iterator, _ATraits, _Alloc > ∗ **null_node_update_pointer**
- typedef base_type::_Iter< node, leaf, head, inode, false > **reverse_iterator**
- typedef __gnu_pbds::detail::synth_access_traits < type_traits, false, access_traits > synth_access_traits

4.324.1 Detailed Description

template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Specialization.

Definition at line 62 of file pat_trie_/traits.hpp.

4.324.2 Member Typedef Documentation

4.324.2.1 template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 88 of file pat_trie_/traits.hpp.

4.324.2.2 template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef Node_Update<**node_const_iterator**, **node_iterator**, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, **pat_trie_tag**, _Alloc >::**node_update**

Type for node update.

Definition at line 93 of file pat_trie_/traits.hpp.

4.324.2.3 template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef **__gnu_pbds::detail::synth_access_traits**<**type_traits**, false, access_traits> __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, **pat_trie_tag**, _Alloc >::**synth_access_traits**

Type for synthesized traits.

Definition at line 74 of file pat_trie_/traits.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_/traits.hpp

4.325 __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference

Public Types

- typedef _ATraits **access_traits**
- typedef base_type::_CIter
  < node, leaf, head, inode,
  true > **const_iterator**
- typedef base_type::_CIter
  < node, leaf, head, inode,
  false > **const_reverse_iterator**
- typedef base_type::_Head
  < synth_access_traits,
  metadata > **head**
- typedef base_type::_Inode
  < synth_access_traits,
  metadata > **inode**
- typedef const_iterator **iterator**
- typedef base_type::_Leaf
  < synth_access_traits,
  metadata > **leaf**
- typedef base_type::_Metadata
  < metadata_type, _Alloc > **metadata**
- typedef
  trie_node_metadata_dispatch
  < Key, null_type, _ATraits,
  Node_Update, _Alloc >::type **metadata_type**
- typedef base_type::_Node_base
  < synth_access_traits,
  metadata > **node**

- typedef base_type::_Node_citer
  < node, leaf, head, inode,
  const_iterator, iterator,
  _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update
  < node_const_iterator,
  node_iterator, _ATraits,
  _Alloc > node_update
- typedef null_node_update
  < node_const_iterator,
  node_const_iterator, _ATraits,
  _Alloc > ∗ **null_node_update_pointer**
- typedef const_reverse_iterator **reverse_iterator**
- typedef
  __gnu_pbds::detail::synth_access_traits
  < type_traits, true,
  access_traits > synth_access_traits

4.325.1 Detailed Description

template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Specialization.

Definition at line 109 of file pat_trie_/traits.hpp.

4.325.2 Member Typedef Documentation

4.325.2.1 template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 135 of file pat_trie_/traits.hpp.

4.325.2.2 template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update

Type for node update.

Definition at line 140 of file pat_trie_/traits.hpp.

4.325.2.3 template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc > typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits

Type for synthesized traits.

Definition at line 121 of file pat_trie_/traits.hpp.

The documentation for this struct was generated from the following file:

- pat_trie_/traits.hpp

4.326 __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false > Struct Template Reference

Public Types

- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef __rebind_ma::value_type **mapped_type**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef stored_data < value_type, null_type > **stored_data_type**
- typedef __rebind_va::value_type **value_type**

4.326.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >

Specialization of type_base for the case where the hash value is not stored alongside each value.

Definition at line 114 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.327 __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true > Struct Template Reference

Public Types

- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef __rebind_ma::value_type **mapped_type**
- typedef __rebind_va::pointer **pointer**

- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef stored_data
  < value_type, size_type > **stored_data_type**
- typedef __rebind_va::value_type **value_type**

4.327.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >

Specialization of type_base for the case where the hash value is stored alongside each value.

Definition at line 147 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.328 __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false > Struct Template Reference

Public Types

- typedef __rebind_va::const_pointer **const_pointer**
- typedef
  __rebind_va::const_reference **const_reference**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef
  __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef __rebind_ma::value_type **mapped_type**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef stored_data
  < value_type, null_type > **stored_data_type**
- typedef Key **value_type**

Static Public Attributes

- static null_type **s_null_type**

4.328.1 Detailed Description

template<typename Key, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >

Specialization of type_base for the case where the hash value is not stored alongside each value.

Definition at line 181 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.329 __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true > Struct Template Reference

Public Types

- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef __rebind_ma::value_type **mapped_type**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef stored_data < value_type, size_type > **stored_data_type**
- typedef Key **value_type**

Static Public Attributes

- static null_type **s_null_type**

4.329.1 Detailed Description

template<typename Key, typename _Alloc>struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >

Specialization of type_base for the case where the hash value is stored alongside each value.

Definition at line 220 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.330 __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

Public Types

- typedef type_base< Key, Mapped, _Alloc, Store_Hash > **type**

4.330.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >

Type base dispatch.

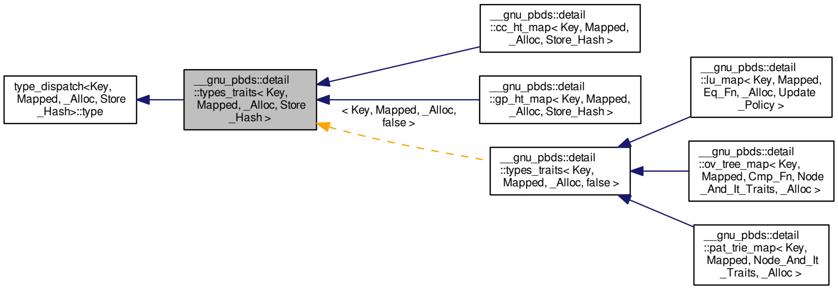
Definition at line 256 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.331 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >:



Public Types

- typedef std::pair< size_type,
  size_type > **comp_hash**
- typedef __rebind_a::const_pointer **key_const_pointer**
- typedef __rebind_a::const_reference **key_const_reference**
- typedef __rebind_a::pointer **key_pointer**
- typedef __rebind_a::reference **key_reference**
- typedef __rebind_a::value_type **key_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int,
  Store_Hash > **store_extra**

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

4.331.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

Traits for abstract types.

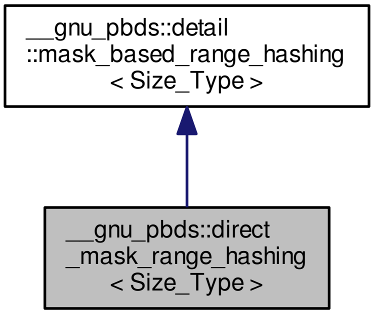
Definition at line 263 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.332 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference

Inheritance diagram for __gnu_pbds::direct_mask_range_hashing< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (direct_mask_range_hashing< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type operator() (size_type hash) const
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

4.332.1 Detailed Description

template<typename Size_Type = std::size_t>class __gnu_pbds::direct_mask_range_hashing< Size_Type >

A mask range-hashing class (uses a bitmask).

Definition at line 109 of file hash_policy.hpp.

4.332.2 Member Function Documentation

4.332.2.1 template<typename Size_Type > direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() (size_type *hash*) const [inline], [protected]

Transforms the __hash value hash into a ranged-hash value (using a bit-mask).

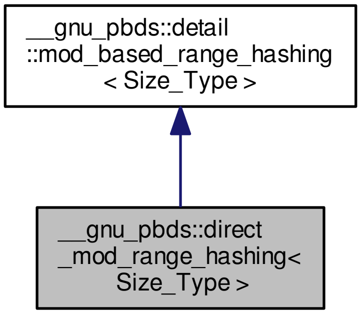
Definition at line 57 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- direct_mask_range_hashing_imp.hpp

4.333 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference

Inheritance diagram for __gnu_pbds::direct_mod_range_hashing< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (direct_mod_range_hashing< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type operator() (size_type hash) const
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

4.333.1 Detailed Description

template<typename Size_Type = std::size_t>class __gnu_pbds::direct_mod_range_hashing< Size_Type >

A mod range-hashing class (uses the modulo function).

Definition at line 141 of file hash_policy.hpp.

4.333.2 Member Function Documentation

4.333.2.1 template<typename Size_Type > direct_mod_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mod_range_hashing< Size_Type >::operator() (size_type *hash*) const `[inline]`, `[protected]`

Transforms the __hash value hash into a ranged-hash value (using a modulo operation).

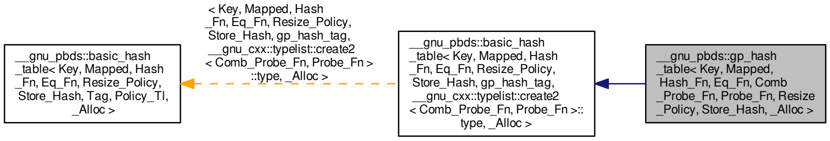
Definition at line 57 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- direct_mod_range_hashing_imp.hpp

4.334 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >:



Public Types

- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef gp_hash_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Probe_Fn **probe_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- gp_hash_table ()
- gp_hash_table (const hash_fn &h)

- gp_hash_table (const hash_fn &h, const eq_fn &e)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- template<typename It >
  gp_hash_table (It first, It last)
- template<typename It >
  gp_hash_table (It first, It last, const hash_fn &h)
- template<typename It >
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It >
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- template<typename It >
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It >
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- **gp_hash_table** (const gp_hash_table &other)
- gp_hash_table & **operator=** (const gp_hash_table &other)
- void **swap** (gp_hash_table &other)

4.334.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >

A general-probing hash-based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Comb_Probe_Fn	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies.
Probe_Fn	Probe functor.
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
_Alloc	Allocator type.

Base tag choices are: gp_hash_tag.

Base is basic_hash_table.

Definition at line 368 of file assoc_container.hpp.

4.334.2 Constructor & Destructor Documentation

4.334.2.1 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ( ) `[inline]`

Default constructor.

Definition at line 382 of file assoc_container.hpp.

4.334.2.2 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & *h*) `[inline]`

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object.

Definition at line 386 of file assoc_container.hpp.

4.334.2.3 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & *h*, const eq_fn & *e*) `[inline]`

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

Definition at line 393 of file assoc_container.hpp.

4.334.2.4 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & *h*, const eq_fn & *e*, const comb_probe_fn & *cp*) `[inline]`

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

Definition at line 401 of file assoc_container.hpp.

4.334.2.5 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & *h,* const eq_fn & *e,* const comb_probe_fn & *cp,* const probe_fn & *p*) `[inline]`

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

Definition at line 410 of file assoc_container.hpp.

4.334.2.6 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (const hash_fn & *h,* const eq_fn & *e,* const comb_probe_fn & *cp,* const probe_fn & *p,* const resize_policy & *rp*) `[inline]`

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the Resize_Policy object of the container object.

Definition at line 422 of file assoc_container.hpp.

4.334.2.7 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It *first,* It *last*) `[inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 430 of file assoc_container.hpp.

4.334.2.8 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It *first,* It *last,* const hash_fn & *h*) `[inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object.

Definition at line 438 of file assoc_container.hpp.

4.334.2.9 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It *first*, It *last*, const hash_fn & *h*, const eq_fn & *e*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

Definition at line 449 of file assoc_container.hpp.

4.334.2.10 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It *first*, It *last*, const hash_fn & *h*, const eq_fn & *e*, const comb_probe_fn & *cp*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

Definition at line 461 of file assoc_container.hpp.

4.334.2.11 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It *first*, It *last*, const hash_fn & *h*, const eq_fn & *e*, const comb_probe_fn & *cp*, const probe_fn & *p*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

Definition at line 475 of file assoc_container.hpp.

4.334.2.12 template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table (It *first*, It *last*, const hash_fn & *h*, const eq_fn & *e*, const comb_probe_fn & *cp*, const probe_fn & *p*, const resize_policy & *rp*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

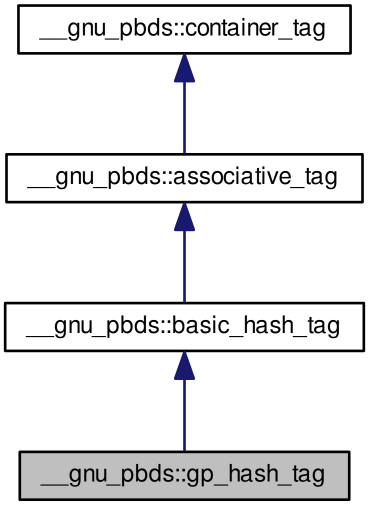
Definition at line 491 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.335 __gnu_pbds::gp_hash_tag Struct Reference

Inheritance diagram for __gnu_pbds::gp_hash_tag:



4.335.1 Detailed Description

General-probing hash.

Definition at line 144 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.336 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- hash_exponential_size_policy (size_type start_size=8, size_type grow_factor=2)
- void **swap** (hash_exponential_size_policy< Size_Type > &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

4.336.1 Detailed Description

template<typename Size_Type = std::size_t>class __gnu_pbds::hash_exponential_size_policy< Size_Type >

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2.

Definition at line 413 of file hash_policy.hpp.

4.336.2 Constructor & Destructor Documentation

4.336.2.1 template<typename Size_Type > __gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (size_type *start_size* = 8, size_type *grow_factor* = 2)

Default constructor, or onstructor taking a start_size, or constructor taking a start size and grow_factor. The policy will use the sequence of sizes start_size, start_size∗ grow_factor, start_size∗ grow_factor^2, ...

Definition at line 44 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- hash_exponential_size_policy_imp.hpp

4.337 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference

Inherits hash_load_check_resize_trigger_size_base< Size_Type, External_Load_Access >.

Public Types

- enum { external_load_access }
- typedef Size_Type **size_type**

Public Member Functions

- hash_load_check_resize_trigger (float load_min=0.125, float load_max=0.5)
- std::pair< float, float > get_loads () const
- void set_loads (std::pair< float, float > load_pair)
- void **swap** (hash_load_check_resize_trigger &other)

Protected Member Functions

- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void notify_cleared ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_entries)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

4.337.1 Detailed Description

template<bool External_Load_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >

A resize trigger policy based on a load check. It keeps the load factor between some load factors load_min and load_max.

Definition at line 175 of file hash_policy.hpp.

4.337.2 Member Enumeration Documentation

4.337.2.1 template<bool External_Load_Access = false, typename Size_Type = std::size_t> anonymous enum

Enumerator:

external_load_access Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

Definition at line 180 of file hash_policy.hpp.

4.337.3 Constructor & Destructor Documentation

4.337.3.1 template<bool External_Load_Access, typename Size_Type > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_resize_trigger (float *load_min* = `0.125`, float *load_max* = `0.5`)

Default constructor, or constructor taking load_min and load_max load factors between which this policy will keep the actual load.

Definition at line 47 of file hash_policy.hpp.

4.337.4 Member Function Documentation

4.337.4.1 template<bool External_Load_Access, typename Size_Type > std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::get_loads ( ) const [inline]

Returns a pair of the minimal and maximal loads, respectively.

Definition at line 236 of file hash_policy.hpp.

4.337.4.2 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared ( ) [protected]

Notifies the table was cleared.

Definition at line 206 of file hash_policy.hpp.

4.337.4.3 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_inserted (size_type *num_entries*) [inline], [protected]

Notifies an element was inserted. The total number of entries in the table is num_entries.

Definition at line 109 of file hash_policy.hpp.

4.337.4.4 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_resized (size_type *new_size*) [protected]

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 151 of file hash_policy.hpp.

4.337.4.5 template<bool External_Load_Access, typename Size_Type > void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (std::pair< float, float > *load_pair*)

Sets the loads through a pair of the minimal and maximal loads, respectively.

Definition at line 245 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- hash_load_check_resize_trigger_imp.hpp

4.338 __gnu_pbds::hash_prime_size_policy Class Reference

Public Types

- typedef std::size_t size_type

Public Member Functions

- hash_prime_size_policy (size_type start_size=8)
- void **swap** (hash_prime_size_policy &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

4.338.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

Definition at line 450 of file hash_policy.hpp.

4.338.2 Member Typedef Documentation

4.338.2.1 typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type

Size type.

Definition at line 454 of file hash_policy.hpp.

4.338.3 Constructor & Destructor Documentation

4.338.3.1 __gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (size_type *start_size* = 8) [inline]

Default constructor, or onstructor taking a start_size The policy will use the sequence of sizes approximately start_size, start_size$*$ 2, start_size$*$ 2^2, ...

Definition at line 127 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- hash_prime_size_policy_imp.hpp

4.339 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference

Inherits Size_Policy, and Trigger_Policy.

Public Types

- enum { **external_size_access** }
- typedef Size_Policy **size_policy**
- typedef Size_Type **size_type**
- typedef Trigger_Policy **trigger_policy**

Public Member Functions

- hash_standard_resize_policy ()
- hash_standard_resize_policy (const Size_Policy &r_size_policy)
- hash_standard_resize_policy (const Size_Policy &r_size_policy, const Trigger_Policy &r_trigger_policy)
- size_type get_actual_size () const

- Size_Policy & get_size_policy ()
- const Size_Policy & get_size_policy () const
- Trigger_Policy & get_trigger_policy ()
- const Trigger_Policy & get_trigger_policy () const
- void resize (size_type suggested_new_size)
- void **swap** (hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > &other)

Protected Member Functions

- size_type get_new_size (size_type size, size_type num_used_e) const
- bool **is_resize_needed** () const
- void **notify_cleared** ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_e)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void **notify_inserted** (size_type num_e)
- void **notify_resized** (size_type new_size)

4.339.1 Detailed Description

template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_check_resize_trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >

A resize policy which delegates operations to size and trigger policies.

Definition at line 489 of file hash_policy.hpp.

4.339.2 Constructor & Destructor Documentation

4.339.2.1 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::hash_standard_resize_policy ( )

Default constructor.

Definition at line 44 of file hash_policy.hpp.

4.339.2.2 template<typename Size_Policy, typename Trigger_Policy , bool External_Size_Access, typename Size_Type > __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::hash_standard_resize_policy (const Size_Policy & *r_size_policy*)

constructor taking some policies r_size_policy will be copied by the Size_Policy object of this object.

Definition at line 50 of file hash_policy.hpp.

4.339.2.3 template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_Type > __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::hash_standard_resize_policy (const Size_Policy & *r_size_policy,* const Trigger_Policy & *r_trigger_policy*)

constructor taking some policies. r_size_policy will be copied by the Size_Policy object of this object. r_trigger_policy will be copied by the Trigger_Policy object of this object.

Definition at line 56 of file hash_policy.hpp.

4.339.3 Member Function Documentation

4.339.3.1 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_actual_size ( ) const `[inline]`

Returns the actual size of the container.

Definition at line 177 of file hash_policy.hpp.

4.339.3.2 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_new_size (size_type *size,* size_type *num_used_e*) const `[protected]`

Queries what the new size should be, when the container is resized naturally. The current __size of the container is size, and the number of used entries within the container is num_used_e.

Definition at line 158 of file hash_policy.hpp.

4.339.3.3 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_size_policy ( )

Access to the Size_Policy object used.

Definition at line 242 of file hash_policy.hpp.

4.339.3.4 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_size_policy ( ) const

Const access to the Size_Policy object used.

Definition at line 248 of file hash_policy.hpp.

4.339.3.5 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy ( )

Access to the Trigger_Policy object used.

Definition at line 230 of file hash_policy.hpp.

4.339.3.6 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::get_trigger_policy ( ) const

Access to the Trigger_Policy object used.

Definition at line 236 of file hash_policy.hpp.

4.339.3.7 template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename Size_Type > void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::resize (size_type *suggested_new_size*)

Resizes the container to suggested_new_size, a suggested size (the actual size will be determined by the Size_Policy object).

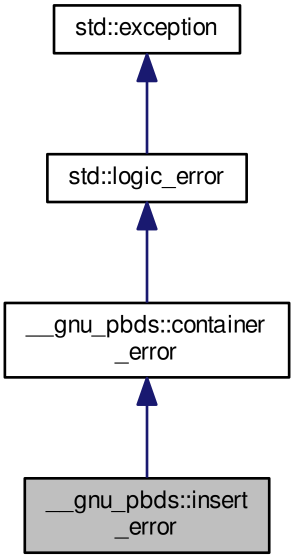
Definition at line 186 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- hash_standard_resize_policy_imp.hpp

4.340 __gnu_pbds::insert_error Struct Reference

Inheritance diagram for __gnu_pbds::insert_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

4.340.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unabvailable, in which case the allocator_type's exception will be thrown).

Definition at line 66 of file exception.hpp.

4.340.2 Member Function Documentation

4.340.2.1 virtual const char∗ std::logic_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

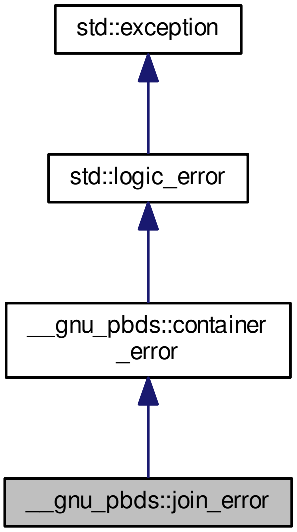
Reimplemented in std::future_error.

The documentation for this struct was generated from the following file:

- exception.hpp

4.341 __gnu_pbds::join_error Struct Reference

Inheritance diagram for __gnu_pbds::join_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

4.341.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

Definition at line 70 of file exception.hpp.

4.341.2 Member Function Documentation

4.341.2.1 virtual const char∗ std::logic_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

Reimplemented in std::future_error.

The documentation for this struct was generated from the following file:

- exception.hpp

4.342 __gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (linear_probe_fn< Size_Type > &other)

Protected Member Functions

- size_type operator() (size_type i) const

4.342.1 Detailed Description

template<typename Size_Type = std::size_t>class __gnu_pbds::linear_probe_fn< Size_Type >

A probe sequence policy using fixed increments.

Definition at line 61 of file hash_policy.hpp.

4.342.2 Member Function Documentation

4.342.2.1 template<typename Size_Type > linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (size_type *i*) const [inline], [protected]

Returns the i-th offset from the hash value.

Definition at line 51 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp

- linear_probe_fn_imp.hpp

4.343 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference

Inherits type< Key, Mapped, _Alloc, list_update_tag, __gnu_cxx::typelist::create2< Eq_Fn, Update_Policy >::type >.

Public Types

- typedef list_update_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Update_Policy **update_policy**

Public Member Functions

- template<typename It >
  list_update (It first, It last)
- **list_update** (const list_update &other)
- list_update & **operator=** (const list_update &other)
- void **swap** (list_update &other)

4.343.1 Detailed Description

template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >

A list-update based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Eq_Fn	Equal functor.
Update_Policy	Update policy, determines when an element will be moved to the front of the list.
_Alloc	Allocator type.

Base is detail::lu_map.

Definition at line 815 of file assoc_container.hpp.

4.343.2 Constructor & Destructor Documentation

4.343.2.1 template<typename Key , typename Mapped , class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (It *first*, It *last*) `[inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

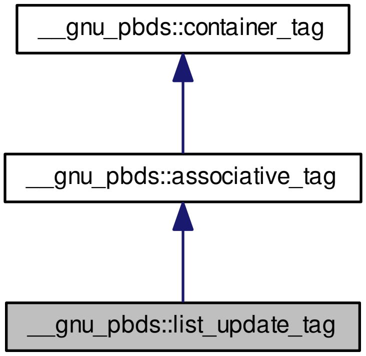
Definition at line 831 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.344 __gnu_pbds::list_update_tag Struct Reference

Inheritance diagram for __gnu_pbds::list_update_tag:



4.344.1 Detailed Description

List-update.

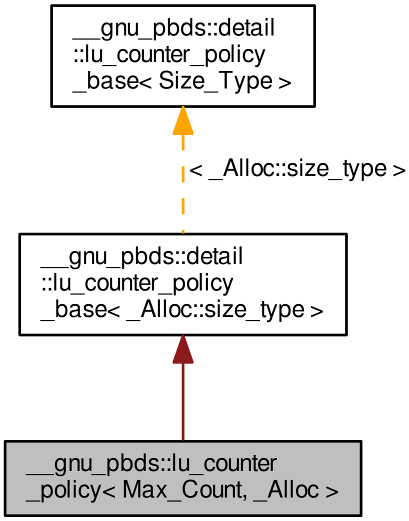
Definition at line 168 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.345 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >:



Public Types

- enum { max_count }
- typedef _Alloc **allocator_type**
- typedef __rebind_m::other::reference metadata_reference
- typedef detail::lu_counter_metadata < size_type > metadata_type
- typedef allocator_type::size_type **size_type**

Public Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference r_data) const

Private Member Functions

- lu_counter_metadata< size_type > **operator()** (size_type max_size) const
- bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const

4.345.1 Detailed Description

template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >

A list-update policy that moves elements to the front of the list based on the counter algorithm.

Definition at line 92 of file list_update_policy.hpp.

4.345.2 Member Typedef Documentation

4.345.2.1 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::metadata_reference

Reference to metadata on which this functor operates.

Definition at line 115 of file list_update_policy.hpp.

4.345.2.2 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> typedef detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::metadata_type

Metadata on which this functor operates.

Definition at line 107 of file list_update_policy.hpp.

4.345.3 Member Enumeration Documentation

4.345.3.1 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> anonymous enum

Enumerator:

max_count When some element is accessed this number of times, it will be moved to the front of the list.

Definition at line 99 of file list_update_policy.hpp.

4.345.4 Member Function Documentation

4.345.4.1 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> metadata_type __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() ( ) const [inline]

Creates a metadata object.

Definition at line 119 of file list_update_policy.hpp.

References __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count.

4.345.4.2 template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>> bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() (metadata_reference *r_data*) const [inline]

Decides whether a metadata object should be moved to the front of the list.

Definition at line 125 of file list_update_policy.hpp.

References __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count.

The documentation for this class was generated from the following file:

- list_update_policy.hpp

4.346 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference

Public Types

- typedef _Alloc **allocator_type**
- typedef
  __rebind_m::other::reference metadata_reference
- typedef null_type metadata_type

Public Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference r_metadata) const

4.346.1 Detailed Description

template<typename _Alloc = std::allocator<char>>class __gnu_pbds::lu_move_to_front_policy< _Alloc >

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

Definition at line 57 of file list_update_policy.hpp.

4.346.2 Member Typedef Documentation

4.346.2.1 template<typename _Alloc = std::allocator<char>> typedef __rebind_m::other::reference __gnu_pbds::lu_move_to_front_policy< _Alloc >::metadata_reference

Reference to metadata on which this functor operates.

Definition at line 70 of file list_update_policy.hpp.

4.346.2.2 template<typename _Alloc = std::allocator<char>> typedef null_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::metadata_type

Metadata on which this functor operates.

Definition at line 63 of file list_update_policy.hpp.

4.346.3 Member Function Documentation

4.346.3.1 template<typename _Alloc = std::allocator<char>> metadata_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() ( ) const `[inline]`

Creates a metadata object.

Definition at line 74 of file list_update_policy.hpp.

4.346.3.2 template<typename _Alloc = std::allocator<char>> bool __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() (metadata_reference *r_metadata*) const [inline]

Decides whether a metadata object should be moved to the front of the list.

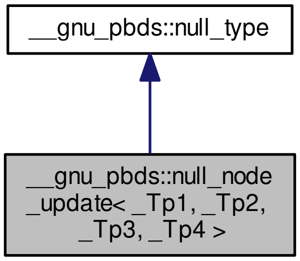
Definition at line 80 of file list_update_policy.hpp.

The documentation for this class was generated from the following file:

- list_update_policy.hpp

4.347 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference

Inheritance diagram for __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >:



4.347.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >

A null node updator, indicating that no node updates are required.

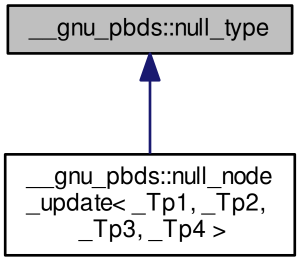
Definition at line 214 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.348 __gnu_pbds::null_type Struct Reference

Inheritance diagram for __gnu_pbds::null_type:



4.348.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

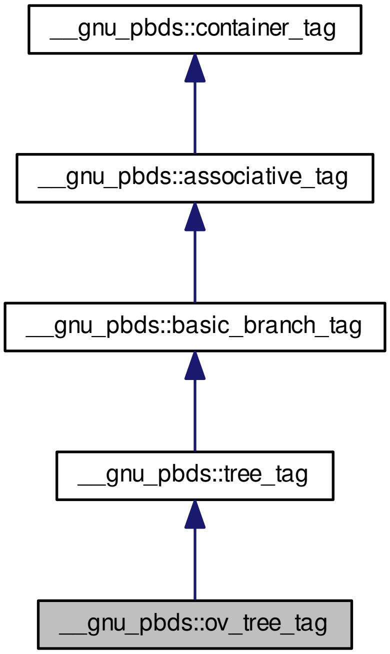
Definition at line 210 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.349 __gnu_pbds::ov_tree_tag Struct Reference

Inheritance diagram for __gnu_pbds::ov_tree_tag:



4.349.1 Detailed Description

Ordered-vector tree.

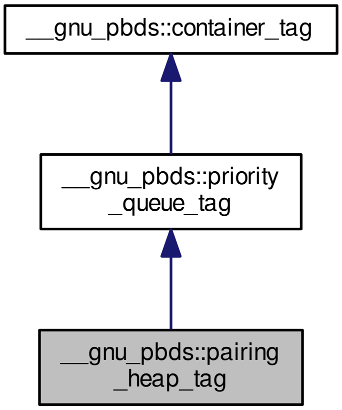
Definition at line 159 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.350 __gnu_pbds::pairing_heap_tag Struct Reference

Inheritance diagram for __gnu_pbds::pairing_heap_tag:



4.350.1 Detailed Description

Pairing-heap.

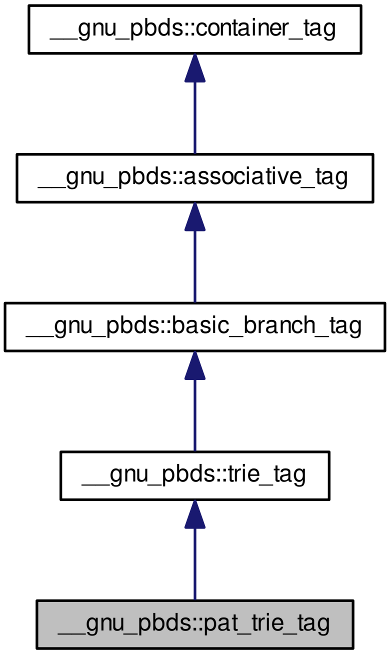
Definition at line 174 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.351 __gnu_pbds::pat_trie_tag Struct Reference

Inheritance diagram for __gnu_pbds::pat_trie_tag:



4.351.1 Detailed Description

PATRICIA trie.

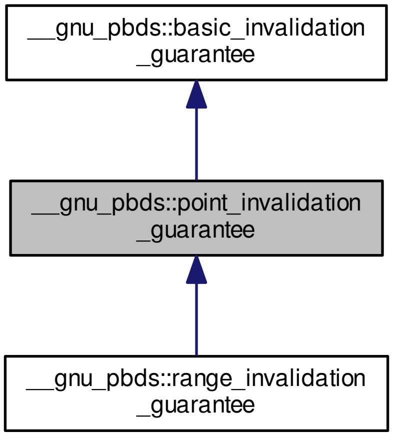
Definition at line 165 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.352 __gnu_pbds::point_invalidation_guarantee Struct Reference

Inheritance diagram for __gnu_pbds::point_invalidation_guarantee:



4.352.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

Definition at line 103 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.353 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference

Inherits type< _Tv, Cmp_Fn, _Alloc, Tag >.

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef Tag **container_category**

- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef _Tv **value_type**

Public Member Functions

- priority_queue (const cmp_fn &r_cmp_fn)
- template<typename It >
  priority_queue (It first_it, It last_it)
- template<typename It >
  priority_queue (It first_it, It last_it, const cmp_fn &r_cmp_fn)
- **priority_queue** (const priority_queue &other)
- priority_queue & **operator=** (const priority_queue &other)
- void **swap** (priority_queue &other)

4.353.1 Detailed Description

template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

A priority queue composed of one specific heap policy.

Template Parameters

_Tv	Value type.
Cmp_Fn	Comparison functor.
Tag	Instantiating data structure type, see container_tag.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: binary_heap_tag, binomial_heap_tag, pairing_heap_tag, rc_binomial_heap_tag, thin_heap_tag

Base choices are: detail::binary_heap, detail::binomial_heap, detail::pairing_heap, detail::rc_binomial_heap, detail::thin_heap.

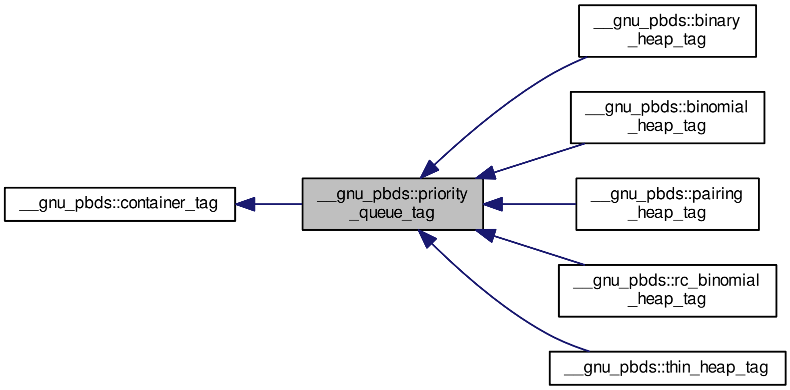
Definition at line 83 of file priority_queue.hpp.

The documentation for this class was generated from the following file:

- priority_queue.hpp

4.354 __gnu_pbds::priority_queue_tag Struct Reference

Inheritance diagram for __gnu_pbds::priority_queue_tag:



4.354.1 Detailed Description

Basic priority-queue.

Definition at line 171 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.355 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (quadratic_probe_fn< Size_Type > &other)

Protected Member Functions

- size_type operator() (size_type i) const

4.355.1 Detailed Description

template<typename Size_Type = std::size_t>class __gnu_pbds::quadratic_probe_fn< Size_Type >

A probe sequence policy using square increments.

Definition at line 85 of file hash_policy.hpp.

4.355.2 Member Function Documentation

4.355.2.1 template<typename Size_Type > quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator() (size_type *i*) const `[inline]`, `[protected]`

Returns the i-th offset from the hash value.

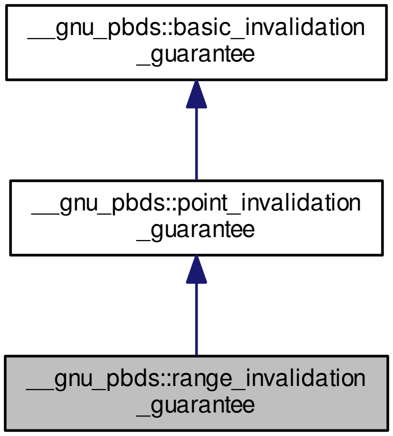
Definition at line 51 of file hash_policy.hpp.

The documentation for this class was generated from the following files:

- hash_policy.hpp
- quadratic_probe_fn_imp.hpp

4.356 __gnu_pbds::range_invalidation_guarantee Struct Reference

Inheritance diagram for __gnu_pbds::range_invalidation_guarantee:



4.356.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of begin() and end()) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

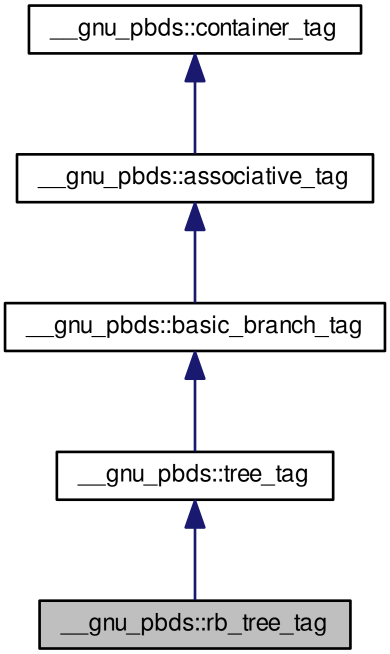
Definition at line 114 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.357 __gnu_pbds::rb_tree_tag Struct Reference

Inheritance diagram for __gnu_pbds::rb_tree_tag:



4.357.1 Detailed Description

Red-black tree.

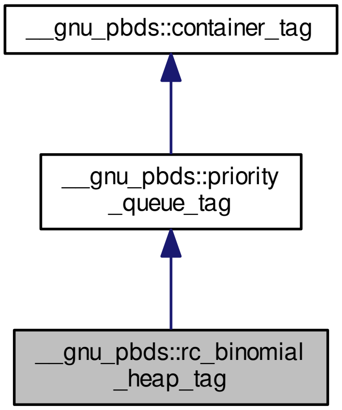
Definition at line 153 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.358 __gnu_pbds::rc_binomial_heap_tag Struct Reference

Inheritance diagram for __gnu_pbds::rc_binomial_heap_tag:



4.358.1 Detailed Description

Redundant-counter binomial-heap.

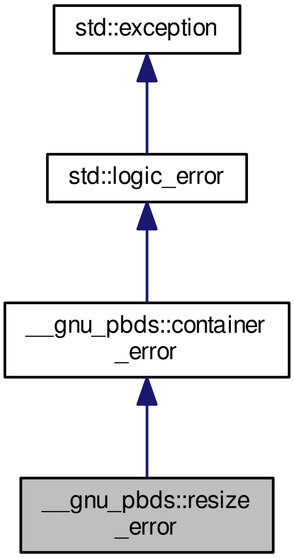
Definition at line 180 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.359 __gnu_pbds::resize_error Struct Reference

Inheritance diagram for __gnu_pbds::resize_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

4.359.1 Detailed Description

A container cannot be resized.

Definition at line 73 of file exception.hpp.

4.359.2 Member Function Documentation

4.359.2.1 virtual const char∗ std::logic_error::what ( ) const `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

Reimplemented in std::future_error.

The documentation for this struct was generated from the following file:

- exception.hpp

4.360 __gnu_pbds::sample_probe_fn Class Reference

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- sample_probe_fn ()
- sample_probe_fn (const sample_probe_fn &)
- void swap (sample_probe_fn &)

Protected Member Functions

- size_type operator() (key_const_reference r_key, size_type i) const

4.360.1 Detailed Description

A sample probe policy.

Definition at line 47 of file sample_probe_fn.hpp.

4.360.2 Constructor & Destructor Documentation

4.360.2.1 __gnu_pbds::sample_probe_fn::sample_probe_fn ( )

Default constructor.

4.360.2.2 __gnu_pbds::sample_probe_fn::sample_probe_fn (const **sample_probe_fn** &)

Copy constructor.

4.360.3 Member Function Documentation

4.360.3.1 size_type __gnu_pbds::sample_probe_fn::operator() (key_const_reference *r_key*, size_type *i*) const `[inline]`, `[protected]`

Returns the i-th offset from the hash value of some key r_key.

4.360.3.2 void __gnu_pbds::sample_probe_fn::swap (**sample_probe_fn** &) `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- sample_probe_fn.hpp

4.361 __gnu_pbds::sample_range_hashing Class Reference

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_range_hashing ()
- sample_range_hashing (const sample_range_hashing &other)
- void swap (sample_range_hashing &other)

Protected Member Functions

- void notify_resized (size_type)
- size_type operator() (size_type) const

4.361.1 Detailed Description

A sample range-hashing functor.

Definition at line 47 of file sample_range_hashing.hpp.

4.361.2 Member Typedef Documentation

4.361.2.1 typedef std::size_t **__gnu_pbds::sample_range_hashing::size_type**

Size type.

Definition at line 51 of file sample_range_hashing.hpp.

4.361.3 Constructor & Destructor Documentation

4.361.3.1 __gnu_pbds::sample_range_hashing::sample_range_hashing ( )

Default constructor.

4.361.3.2 __gnu_pbds::sample_range_hashing::sample_range_hashing (const **sample_range_hashing** & *other*)

Copy constructor.

4.361.4 Member Function Documentation

4.361.4.1 void __gnu_pbds::sample_range_hashing::notify_resized (**size_type**) `[protected]`

Notifies the policy object that the container's size has changed to argument's size.

4.361.4.2 **size_type** __gnu_pbds::sample_range_hashing::operator() (**size_type**) const `[inline]`, `[protected]`

Transforms the __hash value hash into a ranged-hash value.

4.361.4.3 void __gnu_pbds::sample_range_hashing::swap (**sample_range_hashing** & *other*) `[inline]`

Swaps content.

The documentation for this class was generated from the following file:

- sample_range_hashing.hpp

4.362 __gnu_pbds::sample_ranged_hash_fn Class Reference

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- sample_ranged_hash_fn ()
- sample_ranged_hash_fn (const sample_ranged_hash_fn &)
- void swap (sample_ranged_hash_fn &)

Protected Member Functions

- void notify_resized (size_type)
- size_type operator() (key_const_reference) const

4.362.1 Detailed Description

A sample ranged-hash functor.

Definition at line 47 of file sample_ranged_hash_fn.hpp.

4.362.2 Constructor & Destructor Documentation

4.362.2.1 __gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ( )

Default constructor.

4.362.2.2 __gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (const sample_ranged_hash_fn &)

Copy constructor.

4.362.3 Member Function Documentation

4.362.3.1 void __gnu_pbds::sample_ranged_hash_fn::notify_resized (size_type) [protected]

Notifies the policy object that the container's __size has changed to size.

4.362.3.2 size_type __gnu_pbds::sample_ranged_hash_fn::operator() (key_const_reference) const [inline], [protected]

Transforms key_const_reference into a position within the table.

4.362.3.3 void __gnu_pbds::sample_ranged_hash_fn::swap (sample_ranged_hash_fn &) [inline]

Swaps content.

The documentation for this class was generated from the following file:

- sample_ranged_hash_fn.hpp

4.363 __gnu_pbds::sample_ranged_probe_fn Class Reference

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- **sample_ranged_probe_fn** (const sample_ranged_probe_fn &)
- void **swap** (sample_ranged_probe_fn &)

Protected Member Functions

- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference, std::size_t, size_type) const

4.363.1 Detailed Description

A sample ranged-probe functor.

Definition at line 47 of file sample_ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- sample_ranged_probe_fn.hpp

4.364 __gnu_pbds::sample_resize_policy Class Reference

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_resize_policy ()
- sample_range_hashing (const sample_resize_policy &other)
- void swap (sample_resize_policy &other)

Protected Member Functions

- size_type get_new_size (size_type size, size_type num_used_e) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_e)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()

- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_e)
- void notify_resized (size_type new_size)

4.364.1 Detailed Description

A sample resize policy.

Definition at line 47 of file sample_resize_policy.hpp.

4.364.2 Member Typedef Documentation

4.364.2.1 typedef std::size_t __gnu_pbds::sample_resize_policy::size_type

Size type.

Definition at line 51 of file sample_resize_policy.hpp.

4.364.3 Constructor & Destructor Documentation

4.364.3.1 __gnu_pbds::sample_resize_policy::sample_resize_policy ( )

Default constructor.

4.364.4 Member Function Documentation

4.364.4.1 size_type __gnu_pbds::sample_resize_policy::get_new_size (size_type *size*, size_type *num_used_e*) const `[protected]`

Queries what the new size should be.

4.364.4.2 bool __gnu_pbds::sample_resize_policy::is_resize_needed ( ) const `[inline]`, `[protected]`

Queries whether a resize is needed.

4.364.4.3 void __gnu_pbds::sample_resize_policy::notify_cleared ( ) `[protected]`

Notifies the table was cleared.

4.364.4.4 void __gnu_pbds::sample_resize_policy::notify_erase_search_collision ( ) `[inline]`, `[protected]`

Notifies a search encountered a collision.

4.364.4.5 void __gnu_pbds::sample_resize_policy::notify_erase_search_end ( ) `[inline]`, `[protected]`

Notifies a search ended.

4.364.4.6 void __gnu_pbds::sample_resize_policy::notify_erase_search_start ( ) `[inline]`, `[protected]`

Notifies a search started.

4.364.4.7 void __gnu_pbds::sample_resize_policy::notify_erased (**size_type** *num_e*) [inline], [protected]

Notifies an element was erased.

4.364.4.8 void __gnu_pbds::sample_resize_policy::notify_find_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

4.364.4.9 void __gnu_pbds::sample_resize_policy::notify_find_search_end ( ) [inline], [protected]

Notifies a search ended.

4.364.4.10 void __gnu_pbds::sample_resize_policy::notify_find_search_start ( ) [inline], [protected]

Notifies a search started.

4.364.4.11 void __gnu_pbds::sample_resize_policy::notify_insert_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

4.364.4.12 void __gnu_pbds::sample_resize_policy::notify_insert_search_end ( ) [inline], [protected]

Notifies a search ended.

4.364.4.13 void __gnu_pbds::sample_resize_policy::notify_insert_search_start ( ) [inline], [protected]

Notifies a search started.

4.364.4.14 void __gnu_pbds::sample_resize_policy::notify_inserted (**size_type** *num_e*) [inline], [protected]

Notifies an element was inserted.

4.364.4.15 void __gnu_pbds::sample_resize_policy::notify_resized (**size_type** *new_size*) [protected]

Notifies the table was resized to new_size.

4.364.4.16 __gnu_pbds::sample_resize_policy::sample_range_hashing (const **sample_resize_policy** & *other*)

Copy constructor.

4.364.4.17 void __gnu_pbds::sample_resize_policy::swap (**sample_resize_policy** & *other*) [inline]

Swaps content.

The documentation for this class was generated from the following file:

- sample_resize_policy.hpp

4.365 __gnu_pbds::sample_resize_trigger Class Reference

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_resize_trigger ()
- sample_range_hashing (const sample_resize_trigger &)
- void swap (sample_resize_trigger &)

Protected Member Functions

- bool is_grow_needed (size_type size, size_type num_entries) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_entries)
- void notify_externally_resized (size_type new_size)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

4.365.1 Detailed Description

A sample resize trigger policy.

Definition at line 47 of file sample_resize_trigger.hpp.

4.365.2 Member Typedef Documentation

4.365.2.1 typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type

Size type.

Definition at line 51 of file sample_resize_trigger.hpp.

4.365.3 Constructor & Destructor Documentation

4.365.3.1 __gnu_pbds::sample_resize_trigger::sample_resize_trigger ( )

Default constructor.

4.365.4 Member Function Documentation

4.365.4.1 bool __gnu_pbds::sample_resize_trigger::is_grow_needed (size_type *size,* size_type *num_entries*) const [inline], [protected]

Queries whether a grow is needed.

4.365.4.2 bool __gnu_pbds::sample_resize_trigger::is_resize_needed ( ) const [inline], [protected]

Queries whether a resize is needed.

4.365.4.3 void __gnu_pbds::sample_resize_trigger::notify_cleared ( ) [protected]

Notifies the table was cleared.

4.365.4.4 void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

4.365.4.5 void __gnu_pbds::sample_resize_trigger::notify_erase_search_end ( ) [inline], [protected]

Notifies a search ended.

4.365.4.6 void __gnu_pbds::sample_resize_trigger::notify_erase_search_start ( ) [inline], [protected]

Notifies a search started.

4.365.4.7 void __gnu_pbds::sample_resize_trigger::notify_erased (size_type *num_entries*) [inline], [protected]

Notifies an element was erased.

4.365.4.8 void __gnu_pbds::sample_resize_trigger::notify_externally_resized (size_type *new_size*) [protected]

Notifies the table was resized externally.

4.365.4.9 void __gnu_pbds::sample_resize_trigger::notify_find_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

4.365.4.10 void __gnu_pbds::sample_resize_trigger::notify_find_search_end ( ) [inline], [protected]

Notifies a search ended.

4.365.4.11 void __gnu_pbds::sample_resize_trigger::notify_find_search_start ( ) [inline], [protected]

Notifies a search started.

4.365.4.12 void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision ( ) [inline], [protected]

Notifies a search encountered a collision.

4.365.4.13 void __gnu_pbds::sample_resize_trigger::notify_insert_search_end ( ) [inline], [protected]

Notifies a search ended.

4.365.4.14 void __gnu_pbds::sample_resize_trigger::notify_insert_search_start ( ) [inline], [protected]

Notifies a search started.

4.365.4.15 void __gnu_pbds::sample_resize_trigger::notify_inserted (size_type *num_entries*) [inline], [protected]

Notifies an element was inserted. the total number of entries in the table is num_entries.

4.365.4.16 void __gnu_pbds::sample_resize_trigger::notify_resized (**size_type** *new_size*) [protected]

Notifies the table was resized as a result of this object's signifying that a resize is needed.

4.365.4.17 __gnu_pbds::sample_resize_trigger::sample_range_hashing (const **sample_resize_trigger** &)

Copy constructor.

4.365.4.18 void __gnu_pbds::sample_resize_trigger::swap (**sample_resize_trigger** &) [inline]

Swaps content.

The documentation for this class was generated from the following file:

- sample_resize_trigger.hpp

4.366 __gnu_pbds::sample_size_policy Class Reference

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_size_policy ()
- sample_range_hashing (const sample_size_policy &)
- void swap (sample_size_policy &other)

Protected Member Functions

- size_type get_nearest_larger_size (size_type size) const
- size_type get_nearest_smaller_size (size_type size) const

4.366.1 Detailed Description

A sample size policy.

Definition at line 47 of file sample_size_policy.hpp.

4.366.2 Member Typedef Documentation

4.366.2.1 typedef std::size_t __gnu_pbds::sample_size_policy::size_type

Size type.

Definition at line 51 of file sample_size_policy.hpp.

4.366.3 Constructor & Destructor Documentation

4.366.3.1 __gnu_pbds::sample_size_policy::sample_size_policy ( )

Default constructor.

4.366.4 Member Function Documentation

4.366.4.1 size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size (size_type *size*) const [inline], [protected]

Given a __size size, returns a __size that is larger.

4.366.4.2 size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size (size_type *size*) const [inline], [protected]

Given a __size size, returns a __size that is smaller.

4.366.4.3 __gnu_pbds::sample_size_policy::sample_range_hashing (const sample_size_policy &)

Copy constructor.

4.366.4.4 void __gnu_pbds::sample_size_policy::swap (sample_size_policy & *other*) [inline]

Swaps content.

The documentation for this class was generated from the following file:

- sample_size_policy.hpp

4.367 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference

4.367.1 Detailed Description

template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >

A sample node updator.

Definition at line 49 of file sample_tree_node_update.hpp.

The documentation for this class was generated from the following file:

- sample_tree_node_update.hpp

4.368 __gnu_pbds::sample_trie_access_traits Struct Reference

Public Types

- enum { **max_size** }
- typedef _Alloc::template rebind< key_type > **__rebind_k**
- typedef std::string::const_iterator **const_iterator**
- typedef char e_type
- typedef __rebind_k::other::const_reference **key_const_reference**
- typedef std::string **key_type**
- typedef std::size_t **size_type**

Static Public Member Functions

- static const_iterator begin (key_const_reference)
- static size_type e_pos (e_type)
- static const_iterator end (key_const_reference)

4.368.1 Detailed Description

A sample trie element access traits.

Definition at line 47 of file sample_trie_access_traits.hpp.

4.368.2 Member Typedef Documentation

4.368.2.1 typedef char __gnu_pbds::sample_trie_access_traits::e_type

Element type.

Definition at line 57 of file sample_trie_access_traits.hpp.

4.368.3 Member Function Documentation

4.368.3.1 static const_iterator __gnu_pbds::sample_trie_access_traits::begin (key_const_reference) [inline], [static]

Returns a const_iterator to the first element of r_key.

4.368.3.2 static size_type __gnu_pbds::sample_trie_access_traits::e_pos (e_type) [inline], [static]

Maps an element to a position.

4.368.3.3 static const_iterator __gnu_pbds::sample_trie_access_traits::end (key_const_reference) [inline], [static]

Returns a const_iterator to the after-last element of r_key.

The documentation for this struct was generated from the following file:

- sample_trie_access_traits.hpp

4.369 __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

Public Types

- typedef std::size_t **metadata_type**

Protected Member Functions

- sample_trie_node_update ()
- void operator() (node_iterator, node_const_iterator) const

4.369.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

A sample node updator.

Definition at line 49 of file sample_trie_node_update.hpp.

4.369.2 Constructor & Destructor Documentation

4.369.2.1 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_update ( ) `[protected]`

Default constructor.

4.369.3 Member Function Documentation

4.369.3.1 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > void __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator() (node_iterator , node_const_iterator) const `[inline]`, `[protected]`

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

The documentation for this class was generated from the following file:

- sample_trie_node_update.hpp

4.370 __gnu_pbds::sample_update_policy Struct Reference

Public Member Functions

- sample_update_policy ()
- sample_update_policy (const sample_update_policy &)
- void swap (sample_update_policy &other)

Protected Types

- typedef some_metadata_type metadata_type

Protected Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference) const

4.370.1 Detailed Description

A sample list-update policy.

Definition at line 47 of file sample_update_policy.hpp.

4.370.2 Member Typedef Documentation

4.370.2.1 typedef some_metadata_type **__gnu_pbds::sample_update_policy::metadata_type** [protected]

Metadata on which this functor operates.

Definition at line 61 of file sample_update_policy.hpp.

4.370.3 Constructor & Destructor Documentation

4.370.3.1 __gnu_pbds::sample_update_policy::sample_update_policy ( )

Default constructor.

4.370.3.2 __gnu_pbds::sample_update_policy::sample_update_policy (const **sample_update_policy** &)

Copy constructor.

4.370.4 Member Function Documentation

4.370.4.1 **metadata_type** __gnu_pbds::sample_update_policy::operator() ( ) const [protected]

Creates a metadata object.

4.370.4.2 bool __gnu_pbds::sample_update_policy::operator() (metadata_reference) const [protected]

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method shoule return true if the node should be moved to the front of the list.

4.370.4.3 void __gnu_pbds::sample_update_policy::swap (**sample_update_policy** & *other*) [inline]

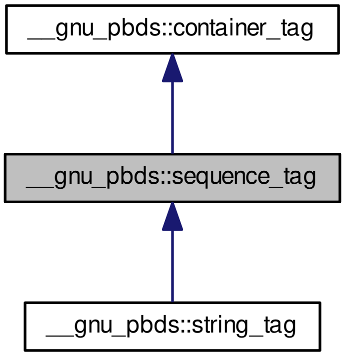
Swaps content.

The documentation for this struct was generated from the following file:

- sample_update_policy.hpp

4.371 __gnu_pbds::sequence_tag Struct Reference

Inheritance diagram for __gnu_pbds::sequence_tag:



4.371.1 Detailed Description

Basic sequence.

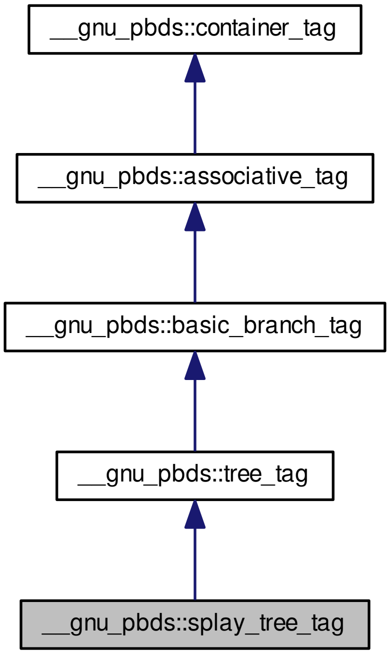
Definition at line 129 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.372 __gnu_pbds::splay_tree_tag Struct Reference

Inheritance diagram for __gnu_pbds::splay_tree_tag:



4.372.1 Detailed Description

Splay tree.

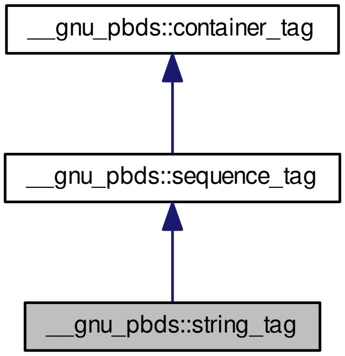
Definition at line 156 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.373 __gnu_pbds::string_tag Struct Reference

Inheritance diagram for __gnu_pbds::string_tag:



4.373.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

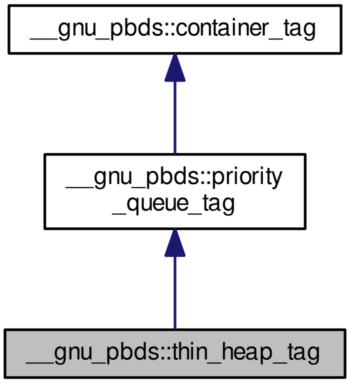
Definition at line 132 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.374 __gnu_pbds::thin_heap_tag Struct Reference

Inheritance diagram for __gnu_pbds::thin_heap_tag:



4.374.1 Detailed Description

Thin heap.

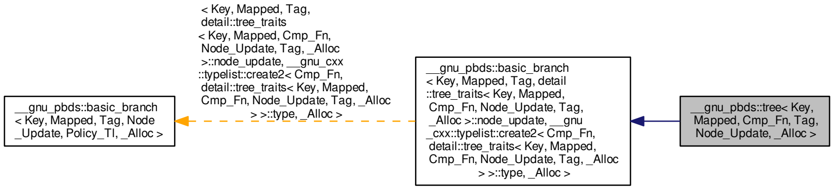
Definition at line 186 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.375 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >:



Public Types

- typedef Cmp_Fn cmp_fn

- typedef detail::tree_traits
  < Key, Mapped, Cmp_Fn,
  Node_Update, Tag, _Alloc >
  ::node_update **node_update**

Public Member Functions

- tree (const cmp_fn &c)
- template< typename It >
  tree (It first, It last)
- template< typename It >
  tree (It first, It last, const cmp_fn &c)
- **tree** (const tree &other)
- tree & **operator=** (const tree &other)
- void **swap** (tree &other)

4.375.1 Detailed Description

template< typename Key, typename Mapped, typename Cmp_Fn = std::less< Key >, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator< char > >class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >

A tree-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Cmp_Fn	Comparison functor.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates tree internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
_Alloc	Allocator type.

Base tag choices are: ov_tree_tag, rb_tree_tag, splay_tree_tag.

Base is basic_branch.

Definition at line 635 of file assoc_container.hpp.

4.375.2 Member Typedef Documentation

4.375.2.1 template< typename Key , typename Mapped , typename Cmp_Fn = std::less< Key >, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator< char > > typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn

Comparison functor type.

Definition at line 642 of file assoc_container.hpp.

4.375.3 Constructor & Destructor Documentation

4.375.3.1 template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (const cmp_fn & *c*) [inline]

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

Definition at line 648 of file assoc_container.hpp.

4.375.3.2 template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (It *first*, It *last*) [inline]

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 655 of file assoc_container.hpp.

4.375.3.3 template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (It *first*, It *last*, const cmp_fn & *c*) [inline]

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.

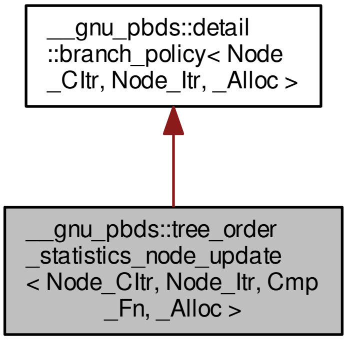
Definition at line 663 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.376 __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef size_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- const_iterator find_by_order (size_type) const
- iterator find_by_order (size_type)
- size_type order_of_key (key_const_reference) const

Protected Member Functions

- void operator() (node_iterator, node_const_iterator) const

Private Types

- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef _Alloc::template rebind< rckey_type >::other **rebind_k**
- typedef _Alloc::template rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Private Member Functions

- virtual it_type **end** ()=0
- it_type **end_iterator** () const

Static Private Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

4.376.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>class __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >

Functor updating ranks of entrees.

Definition at line 64 of file tree_policy.hpp.

4.376.2 Member Function Documentation

4.376.2.1 template<typename Node_CItr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (size_type *order*) const `[inline]`

Finds an entry by __order. Returns a const_iterator to the entry with the __order order, or a const_iterator to the container object's end if order is at least the size of the container object.

Definition at line 72 of file tree_policy.hpp.

4.376.2.2 template<typename Node_CItr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::iterator __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (size_type *order*) `[inline]`

Finds an entry by __order. Returns an iterator to the entry with the __order order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file tree_policy.hpp.

4.376.2.3 template<typename Node_CItr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > void __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::operator() (node_iterator *node_it,* node_const_iterator *end_nd_it*) const `[inline]`, `[protected]`

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

Definition at line 108 of file tree_policy.hpp.

4.376.2.4 template<typename Node_CItr , typename Node_Itr , typename Cmp_Fn , typename _Alloc > tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::size_type __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (key_const_reference *r_key*) const `[inline]`

Returns the order of a key within a sequence. For exapmle, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

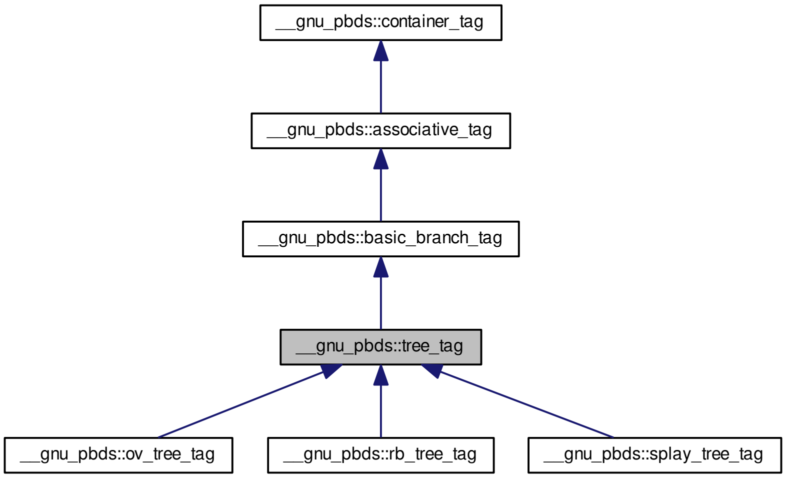
Definition at line 78 of file tree_policy.hpp.

The documentation for this class was generated from the following files:

- tree_policy.hpp
- tree_policy/order_statistics_imp.hpp

4.377 __gnu_pbds::tree_tag Struct Reference

Inheritance diagram for __gnu_pbds::tree_tag:



4.377.1 Detailed Description

Basic tree structure.

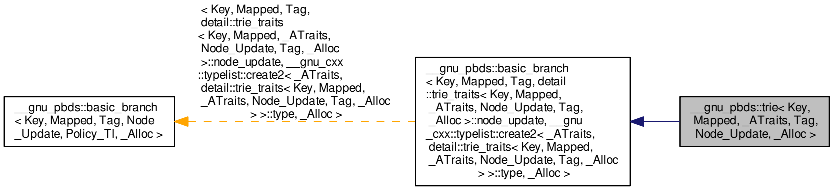
Definition at line 150 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.378 __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >:



Public Types

- typedef _ATraits access_traits

- typedef detail::trie_traits
  < Key, Mapped, _ATraits,
  Node_Update, Tag, _Alloc >
  ::node_update **node_update**

Public Member Functions

- trie (const access_traits &t)
- template<typename It >
  trie (It first, It last)
- template<typename It >
  trie (It first, It last, const access_traits &t)
- **trie** (const trie &other)
- trie & **operator=** (const trie &other)
- void **swap** (trie &other)

4.378.1 Detailed Description

template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

A trie-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
_ATraits	Element access traits.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates trie internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
_Alloc	Allocator type.

Base tag choice is pat_trie_tag.

Base is basic_branch.

Definition at line 731 of file assoc_container.hpp.

4.378.2 Member Typedef Documentation

4.378.2.1 template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<-Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits

Element access traits type.

Definition at line 738 of file assoc_container.hpp.

4.378.3 Constructor & Destructor Documentation

4.378.3.1 template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (const access_traits & *t*) `[inline]`

Constructor taking some policy objects. r_access_traits will be copied by the _ATraits object of the container object.

Definition at line 744 of file assoc_container.hpp.

4.378.3.2 template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (It *first*, It *last*) `[inline]`

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

Definition at line 751 of file assoc_container.hpp.

4.378.3.3 template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>> template<typename It > __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (It *first*, It *last*, const access_traits & *t*) `[inline]`

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object.

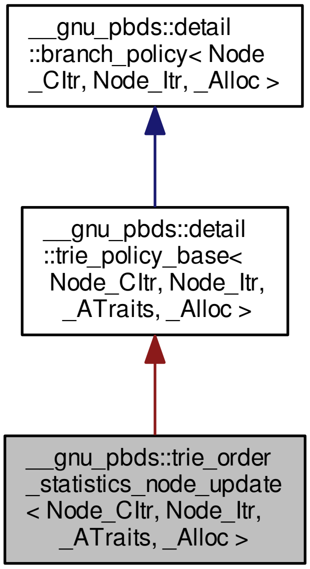
Definition at line 758 of file assoc_container.hpp.

The documentation for this class was generated from the following file:

- assoc_container.hpp

4.379 __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef access_traits::const_iterator **a_const_iterator**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef size_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- const_iterator find_by_order (size_type) const
- iterator find_by_order (size_type)

- size_type order_of_key (key_const_reference) const
- size_type order_of_prefix (a_const_iterator, a_const_iterator) const

Protected Member Functions

- void operator() (node_iterator, node_const_iterator) const

Private Types

- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >
  ::type **rckey_type**
- typedef remove_const
  < value_type >::type **rcvalue_type**
- typedef _Alloc::template
  rebind< rckey_type >::other **rebind_k**
- typedef _Alloc::template
  rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits
  < it_type >::value_type **value_type**

Private Member Functions

- virtual const_iterator **end** () const =0
- it_type **end_iterator** () const
- virtual const access_traits & **get_access_traits** () const =0

Static Private Member Functions

- static size_type **common_prefix_len** (node_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static key_const_reference **extract_key** (const_reference r_val)
- static iterator **leftmost_it** (node_iterator)
- static bool **less** (e_const_iterator, e_const_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static iterator **rightmost_it** (node_iterator)

4.379.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

Functor updating ranks of entrees.

Definition at line 253 of file trie_policy.hpp.

4.379.2 Member Function Documentation

4.379.2.1 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::find_by_order (size_type *order*) const [inline]

Finds an entry by __order. Returns a const_iterator to the entry with the __order order, or a const_iterator to the container object's end if order is at least the size of the container object.

Definition at line 79 of file trie_policy.hpp.

4.379.2.2 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::find_by_order (size_type *order*) [inline]

Finds an entry by __order. Returns an iterator to the entry with the __order order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 45 of file trie_policy.hpp.

4.379.2.3 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > void __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator() (node_iterator *nd_it,* node_const_iterator) const [inline], [protected]

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

Definition at line 152 of file trie_policy.hpp.

4.379.2.4 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_key (key_const_reference *r_key*) const [inline]

Returns the order of a key within a sequence. For exapmle, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

Definition at line 85 of file trie_policy.hpp.

4.379.2.5 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (a_const_iterator *b,* a_const_iterator *e*) const [inline]

Returns the order of a prefix within a sequence. For exapmle, if [b, e] is the smallest prefix, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

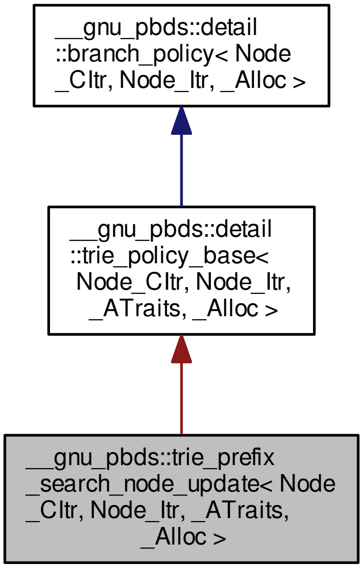
Definition at line 96 of file trie_policy.hpp.

The documentation for this class was generated from the following files:

- trie_policy.hpp
- trie_policy/order_statistics_imp.hpp

4.380 __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

Inheritance diagram for __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef access_traits::const_iterator a_const_iterator
- typedef _ATraits access_traits
- typedef _Alloc allocator_type
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef null_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type size_type

Public Member Functions

- std::pair< const_iterator, const_iterator > prefix_range (key_const_reference) const

- std::pair< iterator, iterator > prefix_range (key_const_reference)
- std::pair< const_iterator,
  const_iterator > prefix_range (a_const_iterator, a_const_iterator) const
- std::pair< iterator, iterator > prefix_range (a_const_iterator, a_const_iterator)

Protected Member Functions

- void operator() (node_iterator node_it, node_const_iterator end_nd_it) const

Private Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >
  ::type **rckey_type**
- typedef remove_const
  < value_type >::type **rcvalue_type**
- typedef _Alloc::template
  rebind< rckey_type >::other **rebind_k**
- typedef _Alloc::template
  rebind< rcvalue_type >::other **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits
  < it_type >::value_type **value_type**

Private Member Functions

- it_type **end_iterator** () const

Static Private Member Functions

- static size_type **common_prefix_len** (node_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static key_const_reference **extract_key** (const_reference r_val)
- static iterator **leftmost_it** (node_iterator)
- static bool **less** (e_const_iterator, e_const_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static iterator **rightmost_it** (node_iterator)

4.380.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>class __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

A node updator that allows tries to be searched for the range of values that match a certain prefix.

Definition at line 155 of file trie_policy.hpp.

4.380.2 Member Typedef Documentation

4.380.2.1 template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef access_traits::const_iterator **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >**::a_const_iterator**

Const element iterator.

Definition at line 168 of file trie_policy.hpp.

4.380.2.2 template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _ATraits **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >**::access_traits**

Element access traits.

Definition at line 165 of file trie_policy.hpp.

4.380.2.3 template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef _Alloc **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >**::allocator_type**

_Alloc type.

Definition at line 171 of file trie_policy.hpp.

4.380.2.4 template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc> typedef allocator_type::size_type **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >**::size_type**

Size type.

Definition at line 174 of file trie_policy.hpp.

4.380.3 Member Function Documentation

4.380.3.1 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > void **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator() (node_iterator *node_it,* node_const_iterator *end_nd_it*) const `[inline]`, `[protected]`

Called to update a node's metadata.

Definition at line 139 of file trie_policy.hpp.

4.380.3.2 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > **std::pair**< typename **trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator, typename **trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator > **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::prefix_range (key_const_reference *r_key*) const

Finds the const iterator range corresponding to all values whose prefixes match r_key.

Definition at line 47 of file trie_policy.hpp.

4.380.3.3 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > **std::pair**< typename **trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator, typename **trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator > **__gnu_pbds::trie_prefix_search_node_update**< Node_CItr, Node_Itr, _ATraits, _Alloc >::prefix_range (key_const_reference *r_key*)

Finds the iterator range corresponding to all values whose prefixes match r_key.

Definition at line 58 of file trie_policy.hpp.

4.380.3.4 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::prefix_range (a_const_iterator *b,* a_const_iterator *e*) const

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 69 of file trie_policy.hpp.

4.380.3.5 template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc > std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::prefix_range (a_const_iterator *b,* a_const_iterator *e*)

Finds the iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 84 of file trie_policy.hpp.

The documentation for this class was generated from the following files:

- trie_policy.hpp
- prefix_search_node_update_imp.hpp

4.381 __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc > Struct Template Reference

Public Types

- enum { **reverse** }
- enum { **min_e_val**, **max_e_val**, **max_size** }
- typedef _Alloc::template rebind< key_type > **__rebind_k**
- typedef detail::__conditional_type < Reverse, typename String::const_reverse_iterator, typename String::const_iterator > ::__type const_iterator
- typedef std::iterator_traits < const_iterator >::value_type e_type
- typedef __rebind_k::other::const_reference **key_const_reference**
- typedef String **key_type**
- typedef _Alloc::size_type **size_type**

Static Public Member Functions

- static const_iterator begin (key_const_reference)
- static size_type e_pos (e_type e)
- static const_iterator end (key_const_reference)

4.381.1 Detailed Description

template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Element access traits for string types.

Template Parameters

String	String type.
Min_E_Val	Minimal element value.
Max_E_Val	Maximum element value.
Reverse	Reverse iteration should be used. Default: false.
_Alloc	Allocator type.

Definition at line 74 of file trie_policy.hpp.

4.381.2 Member Typedef Documentation

4.381.2.1 template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef detail::__conditional_type<Reverse, typename String::const_reverse_iterator, typename String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator

Element const iterator type.

Definition at line 90 of file trie_policy.hpp.

4.381.2.2 template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type

Element type.

Definition at line 93 of file trie_policy.hpp.

4.381.3 Member Function Documentation

4.381.3.1 template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (key_const_reference *r_key*) `[inline]`, `[static]`

Returns a const_iterator to the first element of key_const_reference agumnet.

Definition at line 57 of file trie_policy.hpp.

4.381.3.2 template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > **trie_string_access_traits**< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type **__gnu_pbds::trie_string_access_traits**< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (**e_type** *e*) `[inline]`, `[static]`

Maps an element to a position.

Definition at line 49 of file trie_policy.hpp.

4.381.3.3 template<typename String , typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc > **trie_string_access_traits**< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::**const_iterator** **__gnu_pbds::trie_string_access_traits**< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (key_const_reference *r_key*) `[inline]`, `[static]`

Returns a const_iterator to the after-last element of key_const_reference argument.

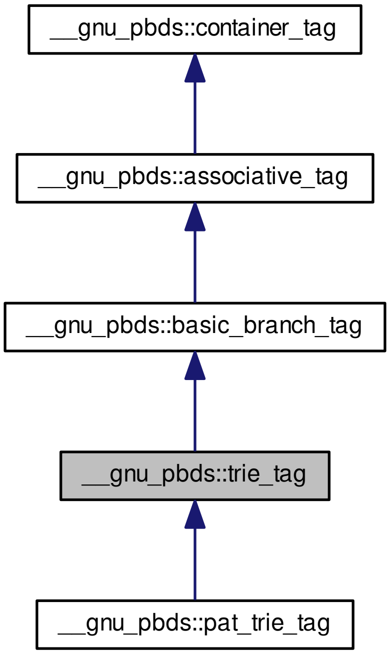
Definition at line 65 of file trie_policy.hpp.

The documentation for this struct was generated from the following files:

- trie_policy.hpp
- trie_string_access_traits_imp.hpp

4.382 __gnu_pbds::trie_tag Struct Reference

Inheritance diagram for __gnu_pbds::trie_tag:



4.382.1 Detailed Description

Basic trie structure.

Definition at line 162 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.383 __gnu_pbds::trivial_iterator_tag Struct Reference

4.383.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.

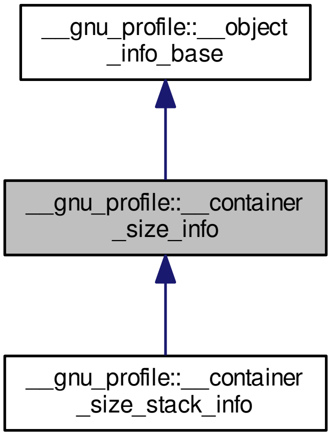
Definition at line 75 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.384 __gnu_profile::__container_size_info Class Reference

Inheritance diagram for __gnu_profile::__container_size_info:



Public Member Functions

- **__container_size_info** (const __container_size_info &__o)
- **__container_size_info** (__stack_t __stack, std::size_t __num)
- std::string **__advice** () const

- void **__destruct** (std::size_t __num, std::size_t __inum)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const __container_size_info &__o)
- void **__resize** (std::size_t __from, std::size_t __to)
- float **__resize_cost** (std::size_t __from, std::size_t)
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**
- bool **_M_valid**

4.384.1 Detailed Description

A container size instrumentation line in the object table.

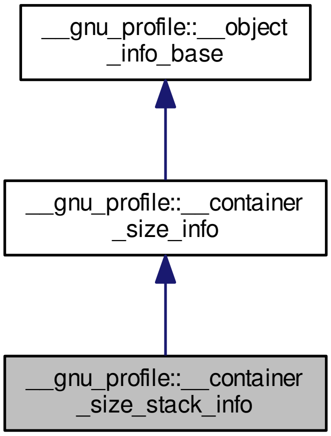
Definition at line 42 of file profiler_container_size.h.

The documentation for this class was generated from the following file:

- profiler_container_size.h

4.385 __gnu_profile::__container_size_stack_info Class Reference

Inheritance diagram for __gnu_profile::__container_size_stack_info:



Public Member Functions

- **__container_size_stack_info** (const __container_size_info &__o)
- std::string **__advice** () const
- void **__destruct** (std::size_t __num, std::size_t __inum)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const __container_size_info &__o)
- void **__resize** (std::size_t __from, std::size_t __to)
- float **__resize_cost** (std::size_t __from, std::size_t)
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**
- bool **_M_valid**

4.385.1 Detailed Description

A container size instrumentation line in the stack table.

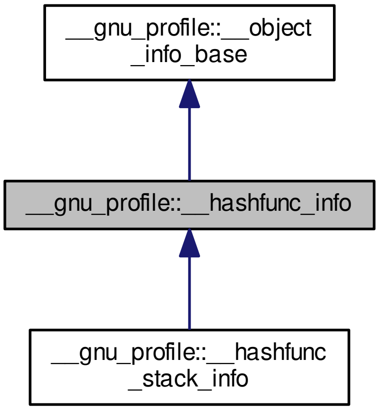
Definition at line 154 of file profiler_container_size.h.

The documentation for this class was generated from the following file:

- profiler_container_size.h

4.386 __gnu_profile::__hashfunc_info Class Reference

Inheritance diagram for __gnu_profile::__hashfunc_info:



Public Member Functions

- **__hashfunc_info** (const __hashfunc_info &__o)
- **__hashfunc_info** (__stack_t __stack)
- std::string **__advice** () const
- void **__destruct** (std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const __hashfunc_info &__o)
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**
- bool **_M_valid**

4.386.1 Detailed Description

A hash performance instrumentation line in the object table.

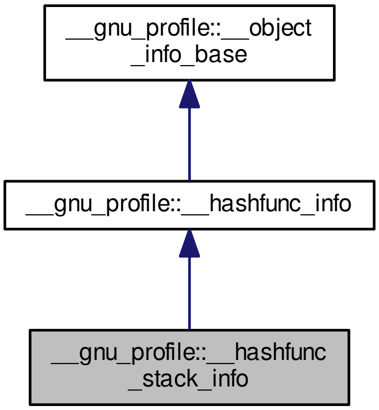
Definition at line 40 of file profiler_hash_func.h.

The documentation for this class was generated from the following file:

- profiler_hash_func.h

4.387 __gnu_profile::__hashfunc_stack_info Class Reference

Inheritance diagram for __gnu_profile::__hashfunc_stack_info:



Public Member Functions

- **__hashfunc_stack_info** (const __hashfunc_info &__o)
- std::string **__advice** () const
- void **__destruct** (std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const __hashfunc_info &__o)
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**
- bool **_M_valid**

4.387.1 Detailed Description

A hash performance instrumentation line in the stack table.

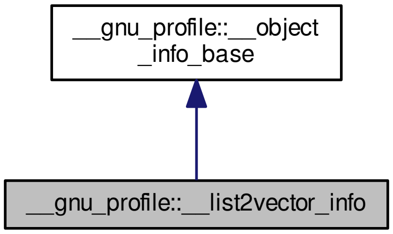
Definition at line 95 of file profiler_hash_func.h.

The documentation for this class was generated from the following file:

- profiler_hash_func.h

4.388 __gnu_profile::__list2vector_info Class Reference

Inheritance diagram for __gnu_profile::__list2vector_info:



Public Member Functions

- **__list2vector_info** (__stack_t __stack)
- **__list2vector_info** (const __list2vector_info &__o)
- std::string **__advice** () const

- bool **__is_valid** ()
- bool **__is_valid** () const
- std::size_t **__iterate** ()
- float **__list_cost** ()
- float **__magnitude** () const
- void **__merge** (const __list2vector_info &__o)
- void **__opr_insert** (std::size_t __shift, std::size_t __size)
- void **__opr_iterate** (std::size_t __num)
- std::size_t **__resize** ()
- void **__resize** (std::size_t __from, std::size_t)
- void **__set_invalid** ()
- void **__set_list_cost** (float __lc)
- void **__set_vector_cost** (float __vc)
- std::size_t **__shift_count** ()
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**

4.388.1 Detailed Description

A list-to-vector instrumentation line in the object table.

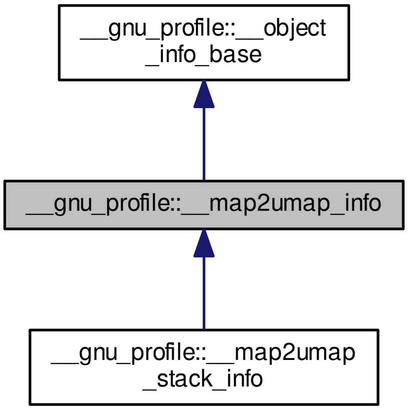
Definition at line 42 of file profiler_list_to_vector.h.

The documentation for this class was generated from the following file:

- profiler_list_to_vector.h

4.389 __gnu_profile::__map2umap_info Class Reference

Inheritance diagram for __gnu_profile::__map2umap_info:



Public Member Functions

- **__map2umap_info** (__stack_t __stack)
- **__map2umap_info** (const __map2umap_info &__o)
- std::string **__advice** () const
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const __map2umap_info &__o)
- void **__record_erase** (std::size_t __size, std::size_t __count)
- void **__record_find** (std::size_t __size)
- void **__record_insert** (std::size_t __size, std::size_t __count)
- void **__record_invalidate** ()
- void **__record_iterate** (std::size_t __count)
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**

4.389.1 Detailed Description

A map-to-unordered_map instrumentation line in the object table.

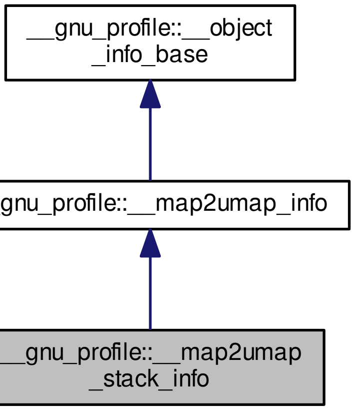
Definition at line 66 of file profiler_map_to_unordered_map.h.

The documentation for this class was generated from the following file:

- profiler_map_to_unordered_map.h

4.390 __gnu_profile::__map2umap_stack_info Class Reference

Inheritance diagram for __gnu_profile::__map2umap_stack_info:



Public Member Functions

- **__map2umap_stack_info** (const __map2umap_info &__o)
- std::string **__advice** () const
- bool **__is_valid** () const
- float **__magnitude** () const
- void **__merge** (const __map2umap_info &__o)
- void **__record_erase** (std::size_t __size, std::size_t __count)
- void **__record_find** (std::size_t __size)
- void **__record_insert** (std::size_t __size, std::size_t __count)
- void **__record_invalidate** ()
- void **__record_iterate** (std::size_t __count)
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**

4.390.1 Detailed Description

A map-to-unordered_map instrumentation line in the stack table.

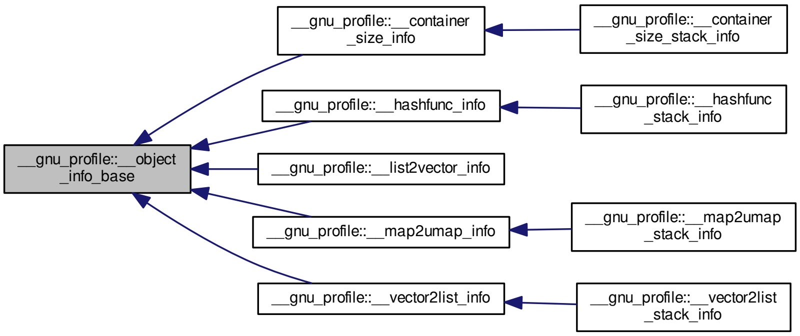
Definition at line 170 of file profiler_map_to_unordered_map.h.

The documentation for this class was generated from the following file:

- profiler_map_to_unordered_map.h

4.391 __gnu_profile::__object_info_base Class Reference

Inheritance diagram for __gnu_profile::__object_info_base:



Public Member Functions

- **__object_info_base** (__stack_t __stack)
- **__object_info_base** (const __object_info_base &__o)
- bool **__is_valid** () const
- __stack_t **__stack** () const
- virtual void **__write** (FILE ∗__f) const =0

Protected Attributes

- __stack_t **_M_stack**
- bool **_M_valid**

4.391.1 Detailed Description

Base class for a line in the object table.

Definition at line 123 of file profiler_node.h.

The documentation for this class was generated from the following file:

- profiler_node.h

4.392 __gnu_profile::__reentrance_guard Struct Reference

Static Public Member Functions

- static bool **__get_in** ()
- static bool & **__inside** ()

4.392.1 Detailed Description

Reentrance guard.

Mechanism to protect all __gnu_profile operations against recursion, multithreaded and exception reentrance.

Definition at line 58 of file profiler.h.

The documentation for this struct was generated from the following file:

- profiler.h

4.393 __gnu_profile::__stack_hash Class Reference

Public Member Functions

- std::size_t **operator()** (__stack_t __s) const
- bool **operator()** (__stack_t __stack1, __stack_t __stack2) const

4.393.1 Detailed Description

Hash function for summary trace using call stack as index.

Definition at line 89 of file profiler_node.h.

The documentation for this class was generated from the following file:

- profiler_node.h

4.394 __gnu_profile::__stack_info_base< __object_info > Class Template Reference

Public Member Functions

- **__stack_info_base** (const __object_info &__info)=0
- virtual const char $*$ **__get_id** () const =0
- virtual float **__magnitude** () const =0
- void **__merge** (const __object_info &__info)=0

4.394.1 Detailed Description

template<typename __object_info>class __gnu_profile::__stack_info_base< __object_info >

Base class for a line in the stack table.

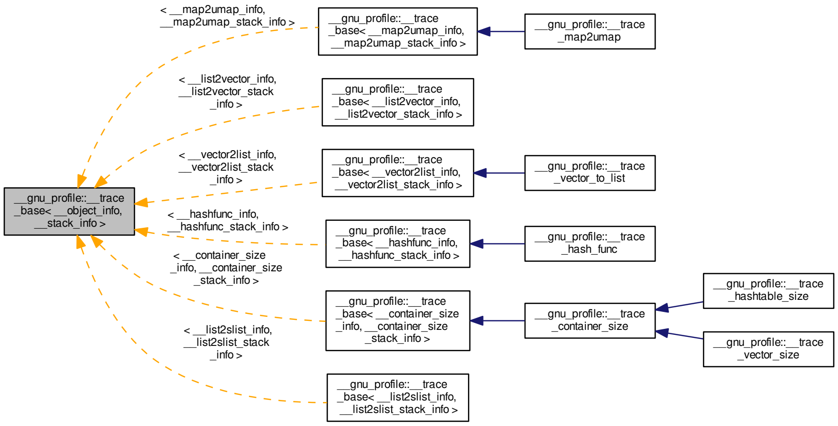
Definition at line 154 of file profiler_node.h.

The documentation for this class was generated from the following file:

- profiler_node.h

4.395 __gnu_profile::__trace_base< __object_info, __stack_info > Class Template Reference

Inheritance diagram for __gnu_profile::__trace_base< __object_info, __stack_info >:



Public Member Functions

- void **__add_object** (__object_t object, __object_info __info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- __object_info ∗ **__get_object_info** (__object_t __object)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.395.1 Detailed Description

template<typename __object_info, typename __stack_info>class __gnu_profile::__trace_base< __object_info, __stack_info >

Base class for all trace producers.

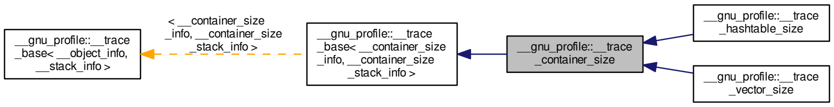
Definition at line 183 of file profiler_trace.h.

The documentation for this class was generated from the following file:

- profiler_trace.h

4.396 __gnu_profile::__trace_container_size Class Reference

Inheritance diagram for __gnu_profile::__trace_container_size:



Public Member Functions

- void **__add_object** (__object_t object, __container_size_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__construct** (const void ∗__obj, std::size_t __inum)
- void **__destruct** (const void ∗__obj, std::size_t __num, std::size_t __inum)
- __container_size_info ∗ **__get_object_info** (__object_t __object)
- void **__insert** (const __object_t __obj, __stack_t __stack, std::size_t __num)
- void **__resize** (const void ∗__obj, int __from, int __to)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.396.1 Detailed Description

Container size instrumentation trace producer.

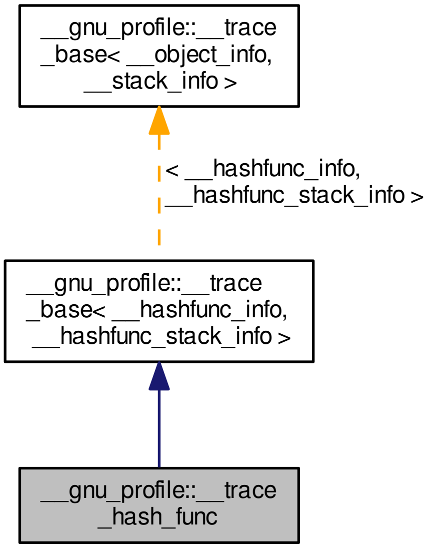
Definition at line 164 of file profiler_container_size.h.

The documentation for this class was generated from the following file:

- profiler_container_size.h

4.397 __gnu_profile::__trace_hash_func Class Reference

Inheritance diagram for __gnu_profile::__trace_hash_func:



Public Member Functions

- void **__add_object** (__object_t object, __hashfunc_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void ∗__obj, std::size_t __chain, std::size_t __accesses, std::size_t __hops)
- __hashfunc_info ∗ **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.397.1 Detailed Description

Hash performance instrumentation producer.

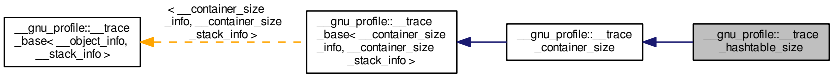
Definition at line 105 of file profiler_hash_func.h.

The documentation for this class was generated from the following file:

- profiler_hash_func.h

4.398 __gnu_profile::__trace_hashtable_size Class Reference

Inheritance diagram for __gnu_profile::__trace_hashtable_size:



Public Member Functions

- void **__add_object** (__object_t object, __container_size_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__construct** (const void ∗__obj, std::size_t __inum)
- void **__destruct** (const void ∗__obj, std::size_t __num, std::size_t __inum)
- __container_size_info ∗ **__get_object_info** (__object_t __object)
- void **__insert** (const __object_t __obj, __stack_t __stack, std::size_t __num)
- void **__resize** (const void ∗__obj, int __from, int __to)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.398.1 Detailed Description

Hashtable size instrumentation trace producer.

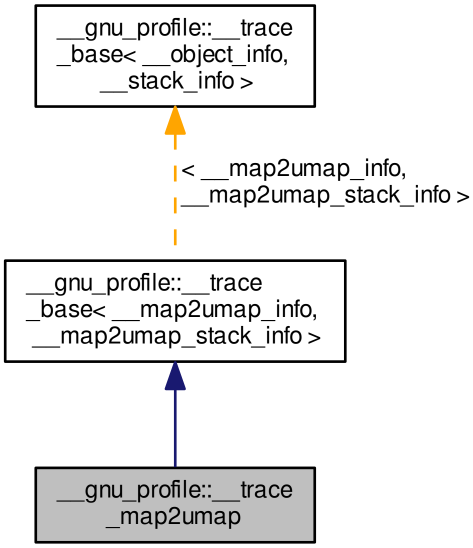
Definition at line 42 of file profiler_hashtable_size.h.

The documentation for this class was generated from the following file:

- profiler_hashtable_size.h

4.399 __gnu_profile::__trace_map2umap Class Reference

Inheritance diagram for __gnu_profile::__trace_map2umap:



Public Member Functions

- void **__add_object** (__object_t object, __map2umap_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- __map2umap_info ∗ **__get_object_info** (__object_t __object)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.399.1 Detailed Description

Map-to-unordered_map instrumentation producer.

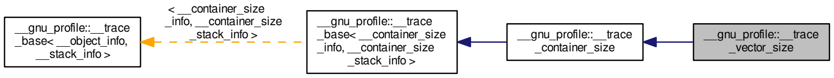
Definition at line 179 of file profiler_map_to_unordered_map.h.

The documentation for this class was generated from the following file:

- profiler_map_to_unordered_map.h

4.400 __gnu_profile::__trace_vector_size Class Reference

Inheritance diagram for __gnu_profile::__trace_vector_size:



Public Member Functions

- void **__add_object** (__object_t object, __container_size_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__construct** (const void ∗__obj, std::size_t __inum)
- void **__destruct** (const void ∗__obj, std::size_t __num, std::size_t __inum)
- __container_size_info ∗ **__get_object_info** (__object_t __object)
- void **__insert** (const __object_t __obj, __stack_t __stack, std::size_t __num)
- void **__resize** (const void ∗__obj, int __from, int __to)
- void **__retire_object** (__object_t __object)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.400.1 Detailed Description

Hashtable size instrumentation trace producer.

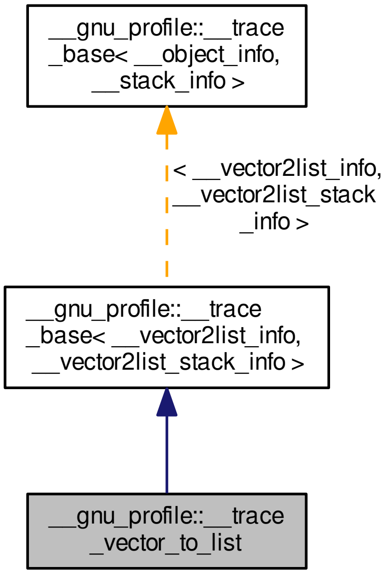
Definition at line 42 of file profiler_vector_size.h.

The documentation for this class was generated from the following file:

- profiler_vector_size.h

4.401 __gnu_profile::__trace_vector_to_list Class Reference

Inheritance diagram for __gnu_profile::__trace_vector_to_list:



Public Member Functions

- void **__add_object** (__object_t object, __vector2list_info__info)
- void **__collect_warnings** (__warning_vector_t &__warnings)
- void **__destruct** (const void ∗__obj)
- __vector2list_info ∗ **__find** (const void ∗__obj)
- __vector2list_info ∗ **__get_object_info** (__object_t __object)
- void **__insert** (__object_t __obj, __stack_t __stack)
- void **__invalid_operator** (const void ∗__obj)
- float **__list_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__opr_find** (const void ∗__obj, std::size_t __size)
- void **__opr_insert** (const void ∗__obj, std::size_t __pos, std::size_t __num)
- void **__opr_iterate** (const void ∗__obj, std::size_t __num)
- void **__resize** (const void ∗__obj, std::size_t __from, std::size_t __to)
- void **__retire_object** (__object_t __object)
- float **__vector_cost** (std::size_t __shift, std::size_t __iterate, std::size_t __resize)
- void **__write** (FILE ∗__f)

Protected Attributes

- const char ∗ **__id**

4.401.1 Detailed Description

Vector-to-list instrumentation producer.

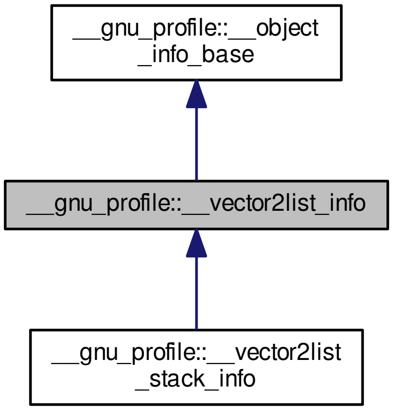
Definition at line 158 of file profiler_vector_to_list.h.

The documentation for this class was generated from the following file:

- profiler_vector_to_list.h

4.402 __gnu_profile::__vector2list_info Class Reference

Inheritance diagram for __gnu_profile::__vector2list_info:



Public Member Functions

- **__vector2list_info** (__stack_t __stack)
- **__vector2list_info** (const __vector2list_info &__o)
- std::string **__advice** () const
- bool **__is_valid** ()
- bool **__is_valid** () const
- std::size_t **__iterate** ()
- float **__list_cost** ()
- float **__magnitude** () const
- void **__merge** (const __vector2list_info &__o)
- void **__opr_find** (std::size_t __size)
- void **__opr_insert** (std::size_t __pos, std::size_t __num)
- void **__opr_iterate** (std::size_t __num)
- std::size_t **__resize** ()
- void **__resize** (std::size_t __from, std::size_t)
- void **__set_invalid** ()

- void **__set_list_cost** (float __lc)
- void **__set_vector_cost** (float __vc)
- std::size_t **__shift_count** ()
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**

4.402.1 Detailed Description

A vector-to-list instrumentation line in the object table.

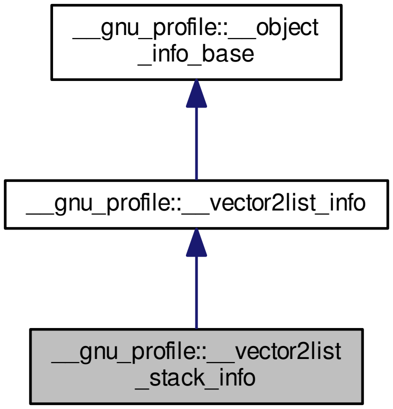
Definition at line 40 of file profiler_vector_to_list.h.

The documentation for this class was generated from the following file:

- profiler_vector_to_list.h

4.403 __gnu_profile::__vector2list_stack_info Class Reference

Inheritance diagram for __gnu_profile::__vector2list_stack_info:



Public Member Functions

- **__vector2list_stack_info** (const __vector2list_info &__o)
- std::string **__advice** () const
- bool **__is_valid** ()
- bool **__is_valid** () const

- std::size_t **__iterate** ()
- float **__list_cost** ()
- float **__magnitude** () const
- void **__merge** (const __vector2list_info &__o)
- void **__opr_find** (std::size_t __size)
- void **__opr_insert** (std::size_t __pos, std::size_t __num)
- void **__opr_iterate** (std::size_t __num)
- std::size_t **__resize** ()
- void **__resize** (std::size_t __from, std::size_t)
- void **__set_invalid** ()
- void **__set_list_cost** (float __lc)
- void **__set_vector_cost** (float __vc)
- std::size_t **__shift_count** ()
- __stack_t **__stack** () const
- void **__write** (FILE ∗__f) const

Protected Attributes

- __stack_t **_M_stack**

4.403.1 Detailed Description

A vector-to-list instrumentation line in the stack table.

Definition at line 148 of file profiler_vector_to_list.h.

The documentation for this class was generated from the following file:

- profiler_vector_to_list.h

4.404 __gnu_profile::__warning_data Struct Reference

Public Member Functions

- **__warning_data** (float __m, __stack_t __c, const char ∗__id, const std::string &__msg)
- bool **operator<** (const __warning_data &__other) const

Public Attributes

- __stack_t **__context**
- float **__magnitude**
- const char ∗ **__warning_id**
- std::string **__warning_message**

4.404.1 Detailed Description

Representation of a warning.

Definition at line 73 of file profiler_trace.h.

The documentation for this struct was generated from the following file:

- profiler_trace.h

4.405 __pool< _Thread > Class Template Reference

4.405.1 Detailed Description

template<bool _Thread>class __pool< _Thread >

Data describing the underlying memory pool, parameterized on threading support.

Definition at line 192 of file mt_allocator.h.

The documentation for this class was generated from the following file:

- mt_allocator.h

4.406 __reflection_typelist< _Elements > Struct Template Reference

4.406.1 Detailed Description

template<typename... _Elements>struct __reflection_typelist< _Elements >

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.

Definition at line 56 of file tr2/type_traits.

The documentation for this struct was generated from the following file:

- tr2/type_traits

4.407 _Bind< _Signature > Struct Template Reference

4.407.1 Detailed Description

template<typename _Signature>struct _Bind< _Signature >

Type of the function object returned from bind().

Definition at line 1277 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.408 _Bind_result< _Result, _Signature > Struct Template Reference

4.408.1 Detailed Description

template<typename _Result, typename _Signature>struct _Bind_result< _Result, _Signature >

Type of the function object returned from bind<R>().

Definition at line 1403 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.409 _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code > Struct Template Reference

4.409.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType, bool __cache_hash_code>struct _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >

Primary class template _Equal_helper.

Definition at line 1156 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.410 _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference

4.410.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>struct _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >

Primary class template _Equality.

This is for implementing equality comparison for unordered containers, per N3068, by John Lakos and Pablo Halpern. Algorithmically, we follow closely the reference implementations therein.

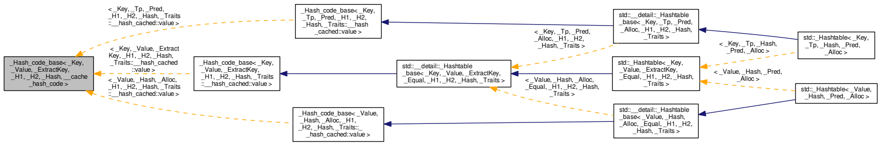
Definition at line 1559 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.411 _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference

Inheritance diagram for _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >:



4.411.1 Detailed Description

template< typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code > struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >

Primary class template _Hash_code_base.

Encapsulates two policy issues that aren't quite orthogonal. (1) the difference between using a ranged hash function and using the combination of a hash function and a range-hashing function. In the former case we don't have such things as hash codes, so we have a dummy type as placeholder. (2) Whether or not we cache hash codes. Caching hash codes is meaningless if we have a ranged hash function.

We also put the key extraction objects here, for convenience. Each specialization derives from one or more of the template parameters to benefit from Ebo. This is important as this type is inherited in some cases by the _Local_iterator_base type used to implement local_iterator and const_local_iterator. As with any iterator type we prefer to make it as small as possible.

Primary template is unused except as a hook for specializations.

Definition at line 907 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.412 _Hash_node< _Value, _Cache_hash_code > Struct Template Reference

4.412.1 Detailed Description

template< typename _Value, bool _Cache_hash_code > struct _Hash_node< _Value, _Cache_hash_code >

Primary template struct _Hash_node.

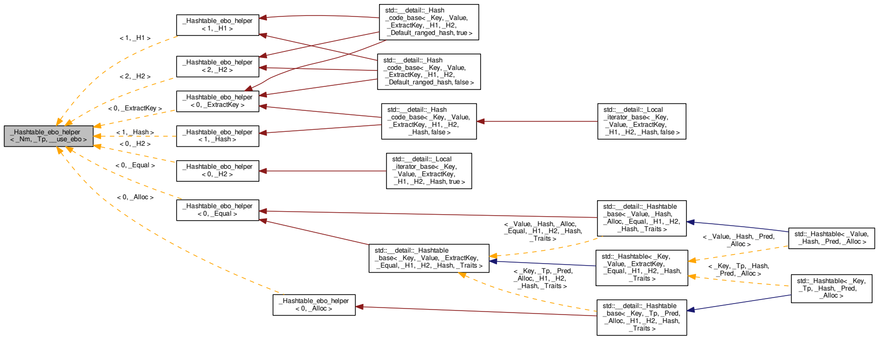
Definition at line 162 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.413 _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo > Struct Template Reference

Inheritance diagram for _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >:



4.413.1 Detailed Description

template<int _Nm, typename _Tp, bool __use_ebo = !__is_final(_Tp) && __is_empty(_Tp)>struct _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >

Primary class template _Hashtable_ebo_helper.

Helper class using EBO when it is not forbidden, type is not final, and when it worth it, type is empty.

Definition at line 831 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.414 _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys > Struct Template Reference

4.414.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Constant_iterators = _Traits::__constant_iterators::value, bool _Unique_keys = _Traits::__unique_keys::value>struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys >

Primary class template _Insert.

Select insert member functions appropriate to _Hashtable policy choices.

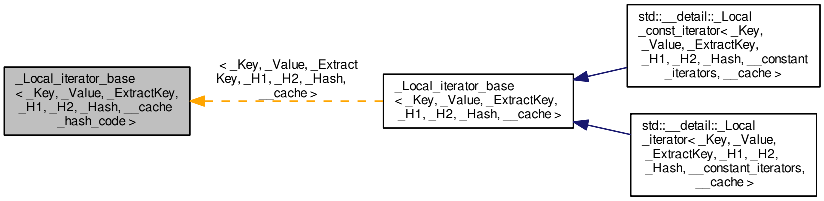
Definition at line 661 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.415 _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference

Inheritance diagram for _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >:



4.415.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code>struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >

Primary class template _Local_iterator_base.

Base class for local iterators, used to iterate within a bucket but not between buckets.

Definition at line 882 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.416 _Mu< _Arg, _IsBindExp, _IsPlaceholder > Class Template Reference

4.416.1 Detailed Description

template<typename _Arg, bool _IsBindExp = is_bind_expression<_Arg>::value, bool _IsPlaceholder = (is_placeholder<_Arg>::value > 0)>class _Mu< _Arg, _IsBindExp, _IsPlaceholder >

Maps an argument to bind() into an actual argument to the bound function object [TR1 3.6.3/5]. Only the first parameter should be specified: the rest are used to determine among the various implementations. Note that, although this class is a function object, it isn't entirely normal because it takes only two parameters regardless of the number of parameters passed to the bind expression. The first parameter is the bound argument and the second parameter is a tuple containing references to the rest of the arguments.

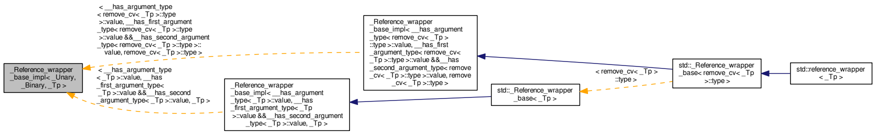
Definition at line 1103 of file functional.

The documentation for this class was generated from the following file:

- functional

4.417 _Reference_wrapper_base_impl< _Unary, _Binary, _Tp > Struct Template Reference

Inheritance diagram for _Reference_wrapper_base_impl< _Unary, _Binary, _Tp >:



4.417.1 Detailed Description

template<bool _Unary, bool _Binary, typename _Tp>struct _Reference_wrapper_base_impl< _Unary, _Binary, _Tp >

Knowing which of unary_function and binary_function _Tp derives from, derives from the same and ensures that reference_wrapper will have a weak result type. See cases below.

Definition at line 269 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.418 _Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference

4.418.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct _Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Primary class template _Rehash_base.

Give hashtable the max_load_factor functions and reserve iff the rehash policy is _Prime_rehash_policy.

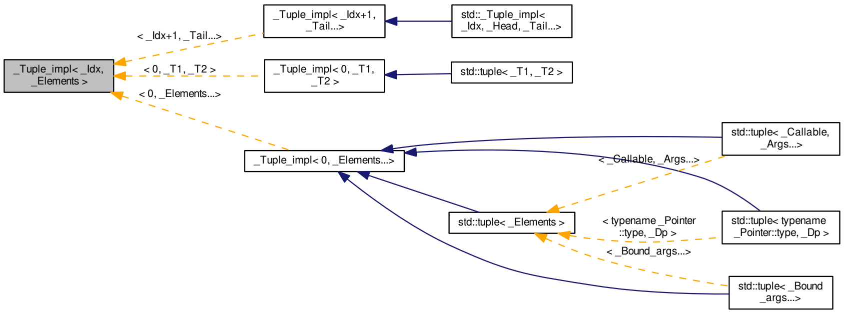
Definition at line 788 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.419 _Tuple_impl< _Idx, _Elements > Struct Template Reference

Inheritance diagram for _Tuple_impl< _Idx, _Elements >:



4.419.1 Detailed Description

template<std::size_t _Idx, typename... _Elements>struct _Tuple_impl< _Idx, _Elements >

Contains the actual implementation of the `tuple` template, stored as a recursive inheritance hierarchy from the first element (most derived class) to the last (least derived class). The `Idx` parameter gives the 0-based index of the element stored at this point in the hierarchy; we use it to implement a constant-time get() operation.

Definition at line 184 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.420 common_type< _Tp > Struct Template Reference

Inherited by std::common_type< _Tp, _Up, _Vp...>.

4.420.1 Detailed Description

template<typename... _Tp>struct common_type< _Tp >

common_type

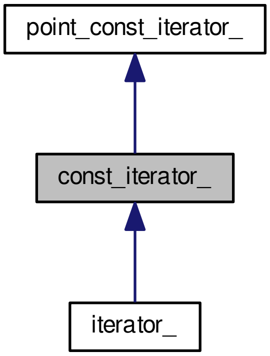
Definition at line 1790 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.421 const_iterator_ Class Reference

Inheritance diagram for const_iterator_:



Public Types

- typedef const_pointer_ const_pointer
- typedef const_reference_ const_reference
- typedef _Alloc::difference_type difference_type
- typedef std::forward_iterator_tag iterator_category
- typedef pointer_ pointer
- typedef reference_ reference
- typedef value_type_ value_type

Public Member Functions

- const_iterator_ ()
- bool operator!= (const point_iterator_ &other) const
- bool operator!= (const point_const_iterator_ &other) const
- const_reference operator$*$ () const

- const_iterator_ & operator++ ()
- const_iterator_ operator++ (int)
- const_pointer operator-> () const
- bool operator== (const point_iterator_ &other) const
- bool operator== (const point_const_iterator_ &other) const

Protected Types

- typedef point_const_iterator_ **base_type**

Protected Member Functions

- const_iterator_ (const_pointer_ p_value, PB_DS_GEN_POS pos, const PB_DS_CLASS_C_DEC ∗p_tbl)

Protected Attributes

- const PB_DS_CLASS_C_DEC ∗ m_p_tbl
- const_pointer **m_p_value**
- PB_DS_GEN_POS **m_pos**

Friends

- class **PB_DS_CLASS_C_DEC**

4.421.1 Detailed Description

Const range-type iterator.

Definition at line 43 of file unordered_iterator/const_iterator.hpp.

4.421.2 Member Typedef Documentation

4.421.2.1 typedef const_pointer_ const_iterator_::const_pointer

Iterator's const pointer type.

Definition at line 60 of file unordered_iterator/const_iterator.hpp.

4.421.2.2 typedef const_reference_ const_iterator_::const_reference

Iterator's const reference type.

Definition at line 66 of file unordered_iterator/const_iterator.hpp.

4.421.2.3 typedef _Alloc::difference_type const_iterator_::difference_type

Difference type.

Definition at line 51 of file unordered_iterator/const_iterator.hpp.

4.421.2.4 typedef std::forward_iterator_tag const_iterator_::iterator_category

Category.

Definition at line 48 of file unordered_iterator/const_iterator.hpp.

4.421.2.5 typedef pointer_ const_iterator_::pointer

Iterator's pointer type.

Definition at line 57 of file unordered_iterator/const_iterator.hpp.

4.421.2.6 typedef reference_ const_iterator_::reference

Iterator's reference type.

Definition at line 63 of file unordered_iterator/const_iterator.hpp.

4.421.2.7 typedef value_type_ const_iterator_::value_type

Iterator's value type.

Definition at line 54 of file unordered_iterator/const_iterator.hpp.

4.421.3 Constructor & Destructor Documentation

4.421.3.1 const_iterator_::const_iterator_ ( ) [inline]

Default constructor.

Definition at line 69 of file unordered_iterator/const_iterator.hpp.

4.421.3.2 const_iterator_::const_iterator_ (const_pointer_ *p_value*, PB_DS_GEN_POS *pos*, const PB_DS_CLASS_C_DEC ∗ *p_tbl*) [inline], [protected]

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a find() of a table.

Definition at line 97 of file unordered_iterator/const_iterator.hpp.

4.421.4 Member Function Documentation

4.421.4.1 bool point_const_iterator_::operator!= (const point_iterator_ & *other*) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered_iterator/point_const_iterator.hpp.

4.421.4.2 bool point_const_iterator_::operator!= (const point_const_iterator_ & *other*) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered_iterator/point_const_iterator.hpp.

4.421.4.3 const_reference point_const_iterator_::operator∗ ( ) const [inline], [inherited]

Access.

Definition at line 100 of file unordered_iterator/point_const_iterator.hpp.

4.421.4.4 const_iterator_& const_iterator_::operator++ ( ) [inline]

Increments.

Definition at line 74 of file unordered_iterator/const_iterator.hpp.

References m_p_tbl.

4.421.4.5 const_iterator_ const_iterator_::operator++ (int) [inline]

Increments.

Definition at line 82 of file unordered_iterator/const_iterator.hpp.

References m_p_tbl.

4.421.4.6 const_pointer point_const_iterator_::operator-> ( ) const [inline], [inherited]

Access.

Definition at line 92 of file unordered_iterator/point_const_iterator.hpp.

4.421.4.7 bool point_const_iterator_::operator== (const point_iterator_ & *other*) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 108 of file unordered_iterator/point_const_iterator.hpp.

4.421.4.8 bool point_const_iterator_::operator== (const point_const_iterator_ & *other*) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 113 of file unordered_iterator/point_const_iterator.hpp.

4.421.5 Member Data Documentation

4.421.5.1 const PB_DS_CLASS_C_DEC∗ const_iterator_::m_p_tbl [protected]

Pointer to the table object which created the iterator (used for incrementing its position.

Definition at line 106 of file unordered_iterator/const_iterator.hpp.

Referenced by operator++(), and iterator_::operator++().

The documentation for this class was generated from the following file:

- unordered_iterator/const_iterator.hpp

4.422 container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl > Struct Template Reference

4.422.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_Tl = null_type>struct container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >

Dispatch mechanism, primary template for associative types.

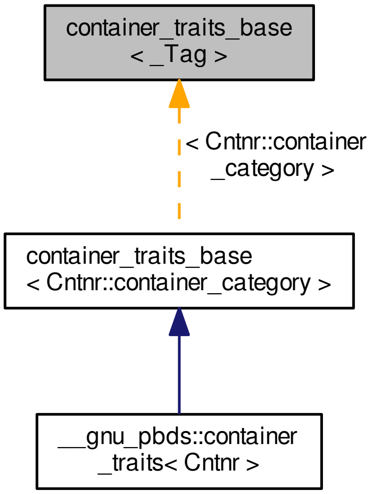
Definition at line 449 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.423 container_traits_base< _Tag > Struct Template Reference

Inheritance diagram for container_traits_base< _Tag >:



4.423.1 Detailed Description

template<typename _Tag>struct container_traits_base< _Tag >

Primary template, container traits base.

Definition at line 220 of file tag_and_trait.hpp.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

4.424 default_trie_access_traits< Key > Struct Template Reference

4.424.1 Detailed Description

template<typename Key>struct default_trie_access_traits< Key >

Primary template, default_trie_access_traits.

Definition at line 135 of file standard_policies.hpp.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

4.425 entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw > Struct Template Reference

4.425.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>struct entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >

Entry compare, primary template.

Definition at line 50 of file entry_cmp.hpp.

The documentation for this struct was generated from the following file:

- entry_cmp.hpp

4.426 entry_pred< _VTp, Pred, _Alloc, No_Throw > Struct Template Reference

4.426.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>struct entry_pred< _VTp, Pred, _Alloc, No_Throw >

Entry predicate primary class template.

Definition at line 50 of file entry_pred.hpp.

The documentation for this struct was generated from the following file:

- entry_pred.hpp

4.427 hash< _Tp > Struct Template Reference

4.427.1 Detailed Description

template<typename _Tp>struct hash< _Tp >

Primary class template hash.

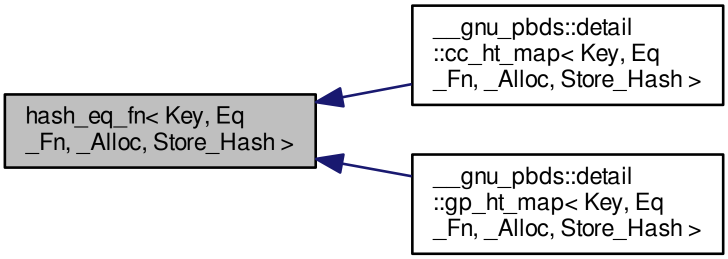
Definition at line 58 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.428 hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference

Inheritance diagram for hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >:



4.428.1 Detailed Description

template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>struct hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >

Primary template.

Definition at line 54 of file hash_eq_fn.hpp.

The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

4.429 hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference

4.429.1 Detailed Description

template<typename Size_Type, bool Hold_Size>class hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >

Primary template.

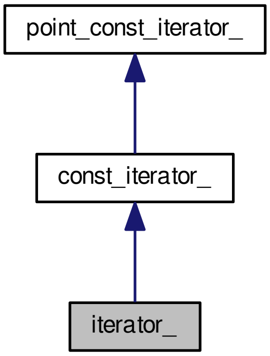
Definition at line 50 of file hash_load_check_resize_trigger_size_base.hpp.

The documentation for this class was generated from the following file:

- hash_load_check_resize_trigger_size_base.hpp

4.430 iterator_ Class Reference

Inheritance diagram for iterator_:



Public Types

- typedef const_pointer_ const_pointer
- typedef const_reference_ const_reference
- typedef _Alloc::difference_type difference_type
- typedef std::forward_iterator_tag iterator_category
- typedef pointer_ pointer
- typedef reference_ reference
- typedef value_type_ value_type

Public Member Functions

- iterator_ ()
- operator const point_iterator_ () const
- operator point_iterator_ ()
- bool operator!= (const point_iterator_ &other) const
- bool operator!= (const point_const_iterator_ &other) const
- reference operator∗ () const
- iterator_ & operator++ ()
- iterator_ operator++ (int)
- pointer operator-> () const
- bool operator== (const point_iterator_ &other) const
- bool operator== (const point_const_iterator_ &other) const

Protected Types

- typedef const_iterator_ **base_type**

Protected Member Functions

- iterator_ (pointer p_value, PB_DS_GEN_POS pos, PB_DS_CLASS_C_DEC ∗p_tbl)

Protected Attributes

- const PB_DS_CLASS_C_DEC ∗ m_p_tbl
- const_pointer **m_p_value**
- PB_DS_GEN_POS **m_pos**

Friends

- class **PB_DS_CLASS_C_DEC**

4.430.1 Detailed Description

Range-type iterator.

Definition at line 43 of file iterator.hpp.

4.430.2 Member Typedef Documentation

4.430.2.1 typedef const_pointer_ iterator_::const_pointer

Iterator's const pointer type.

Definition at line 60 of file iterator.hpp.

4.430.2.2 typedef const_reference_ iterator_::const_reference

Iterator's const reference type.

Definition at line 66 of file iterator.hpp.

4.430.2.3 typedef _Alloc::difference_type iterator_::difference_type

Difference type.

Definition at line 51 of file iterator.hpp.

4.430.2.4 typedef std::forward_iterator_tag iterator_::iterator_category

Category.

Definition at line 48 of file iterator.hpp.

4.430.2.5 typedef pointer_ iterator_::pointer

Iterator's pointer type.

Definition at line 57 of file iterator.hpp.

4.430.2.6 typedef reference_ iterator_::reference

Iterator's reference type.

Definition at line 63 of file iterator.hpp.

4.430.2.7 typedef value_type_ iterator_::value_type

Iterator's value type.

Definition at line 54 of file iterator.hpp.

4.430.3 Constructor & Destructor Documentation

4.430.3.1 iterator_::iterator_ ( ) [inline]

Default constructor.

Definition at line 70 of file iterator.hpp.

4.430.3.2 iterator_::iterator_ (pointer *p_value*, PB_DS_GEN_POS *pos*, PB_DS_CLASS_C_DEC * *p_tbl*) [inline], [protected]

Constructor used by the table to initiate the generalized pointer and position (e.g., this is called from within a find() of a table.

Definition at line 125 of file iterator.hpp.

4.430.4 Member Function Documentation

4.430.4.1 iterator_::operator const point_iterator_ ( ) const [inline]

Conversion to a point-type iterator.

Definition at line 80 of file iterator.hpp.

4.430.4.2 iterator_::operator point_iterator_ ( ) [inline]

Conversion to a point-type iterator.

Definition at line 75 of file iterator.hpp.

4.430.4.3 bool point_const_iterator_::operator!= (const point_iterator_ & *other*) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered_iterator/point_const_iterator.hpp.

4.430.4.4 bool point_const_iterator_::operator!= (const point_const_iterator_ & *other*) const [inline], [inherited]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered_iterator/point_const_iterator.hpp.

4.430.4.5 reference iterator_::operator* ( ) const [inline]

Access.

Definition at line 93 of file iterator.hpp.

4.430.4.6 iterator_& iterator_::operator++ ( ) [inline]

Increments.

Definition at line 101 of file iterator.hpp.

References const_iterator_::m_p_tbl.

4.430.4.7 **iterator_** iterator_::operator++ (int) [inline]

Increments.

Definition at line 109 of file iterator.hpp.

References const_iterator_::m_p_tbl.

4.430.4.8 **pointer** iterator_::operator-> ( ) const [inline]

Access.

Definition at line 85 of file iterator.hpp.

4.430.4.9 bool point_const_iterator_::operator== (const **point_iterator_** & *other*) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 108 of file unordered_iterator/point_const_iterator.hpp.

4.430.4.10 bool point_const_iterator_::operator== (const **point_const_iterator_** & *other*) const [inline], [inherited]

Compares content to a different iterator object.

Definition at line 113 of file unordered_iterator/point_const_iterator.hpp.

4.430.5 Member Data Documentation

4.430.5.1 const PB_DS_CLASS_C_DEC* const_iterator_::m_p_tbl [protected], [inherited]

Pointer to the table object which created the iterator (used for incrementing its position.

Definition at line 106 of file unordered_iterator/const_iterator.hpp.

Referenced by const_iterator_::operator++(), and operator++().

The documentation for this class was generated from the following file:

- iterator.hpp

4.431 owner_less< _Tp > Struct Template Reference

4.431.1 Detailed Description

template<typename _Tp>struct owner_less< _Tp >

Primary template owner_less.

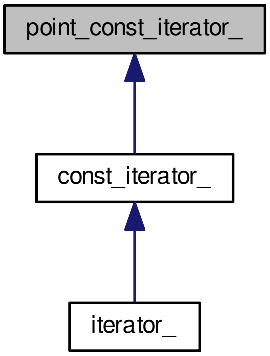
Definition at line 523 of file shared_ptr.h.

The documentation for this struct was generated from the following file:

- shared_ptr.h

4.432 point_const_iterator_ Class Reference

Inheritance diagram for point_const_iterator_:



Public Types

- typedef const_pointer_ const_pointer
- typedef const_reference_ const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef pointer_ pointer
- typedef reference_ reference
- typedef value_type_ value_type

Public Member Functions

- **point_const_iterator_** (const_pointer p_value)
- point_const_iterator_ ()
- point_const_iterator_ (const point_const_iterator_ &other)
- point_const_iterator_ (const point_iterator_ &other)
- bool operator!= (const point_iterator_ &other) const
- bool operator!= (const point_const_iterator_ &other) const
- const_reference operator* () const
- const_pointer operator-> () const
- bool operator== (const point_iterator_ &other) const
- bool operator== (const point_const_iterator_ &other) const

Protected Attributes

- const_pointer **m_p_value**

Friends

- class **PB_DS_CLASS_C_DEC**
- class **point_iterator_**

4.432.1 Detailed Description

Const point-type iterator.

Definition at line 45 of file unordered_iterator/point_const_iterator.hpp.

4.432.2 Member Typedef Documentation

4.432.2.1 typedef const_pointer_ point_const_iterator_::const_pointer

Iterator's const pointer type.

Definition at line 61 of file unordered_iterator/point_const_iterator.hpp.

4.432.2.2 typedef const_reference_ point_const_iterator_::const_reference

Iterator's const reference type.

Definition at line 67 of file unordered_iterator/point_const_iterator.hpp.

4.432.2.3 typedef trivial_iterator_difference_type point_const_iterator_::difference_type

Difference type.

Definition at line 52 of file unordered_iterator/point_const_iterator.hpp.

4.432.2.4 typedef trivial_iterator_tag point_const_iterator_::iterator_category

Category.

Definition at line 49 of file unordered_iterator/point_const_iterator.hpp.

4.432.2.5 typedef pointer_ point_const_iterator_::pointer

Iterator's pointer type.

Definition at line 58 of file unordered_iterator/point_const_iterator.hpp.

4.432.2.6 typedef reference_ point_const_iterator_::reference

Iterator's reference type.

Definition at line 64 of file unordered_iterator/point_const_iterator.hpp.

4.432.2.7 typedef value_type_ point_const_iterator_::value_type

Iterator's value type.

Definition at line 55 of file unordered_iterator/point_const_iterator.hpp.

4.432.3 Constructor & Destructor Documentation

4.432.3.1 point_const_iterator_::point_const_iterator_ ( ) [inline]

Default constructor.

Definition at line 75 of file unordered_iterator/point_const_iterator.hpp.

4.432.3.2 point_const_iterator_::point_const_iterator_ (const **point_const_iterator_** & *other*) [inline]

Copy constructor.

Definition at line 80 of file unordered_iterator/point_const_iterator.hpp.

4.432.3.3 point_const_iterator_::point_const_iterator_ (const **point_iterator_** & *other*) [inline]

Copy constructor.

Definition at line 86 of file unordered_iterator/point_const_iterator.hpp.

4.432.4 Member Function Documentation

4.432.4.1 bool point_const_iterator_::operator!= (const **point_iterator_** & *other*) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 118 of file unordered_iterator/point_const_iterator.hpp.

4.432.4.2 bool point_const_iterator_::operator!= (const **point_const_iterator_** & *other*) const [inline]

Compares content (negatively) to a different iterator object.

Definition at line 123 of file unordered_iterator/point_const_iterator.hpp.

4.432.4.3 **const_reference** point_const_iterator_::operator* ( ) const [inline]

Access.

Definition at line 100 of file unordered_iterator/point_const_iterator.hpp.

4.432.4.4 **const_pointer** point_const_iterator_::operator-> ( ) const [inline]

Access.

Definition at line 92 of file unordered_iterator/point_const_iterator.hpp.

4.432.4.5 bool point_const_iterator_::operator== (const **point_iterator_** & *other*) const [inline]

Compares content to a different iterator object.

Definition at line 108 of file unordered_iterator/point_const_iterator.hpp.

4.432.4.6 bool point_const_iterator_::operator== (const **point_const_iterator_** & *other*) const [inline]

Compares content to a different iterator object.

Definition at line 113 of file unordered_iterator/point_const_iterator.hpp.

The documentation for this class was generated from the following file:

- unordered_iterator/point_const_iterator.hpp

4.433 point_iterator_ Class Reference

Public Types

- typedef const_pointer_ const_pointer
- typedef const_reference_ const_reference
- typedef
  trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef pointer_ pointer
- typedef reference_ reference
- typedef value_type_ value_type

Public Member Functions

- point_iterator_ ()
- point_iterator_ (const point_iterator_ &other)
- **point_iterator_** (pointer p_value)
- bool operator!= (const point_iterator_ &other) const
- bool operator!= (const point_const_iterator_ &other) const
- reference operator* () const
- pointer operator-> () const
- bool operator== (const point_iterator_ &other) const
- bool operator== (const point_const_iterator_ &other) const

Protected Attributes

- pointer **m_p_value**

Friends

- class **PB_DS_CLASS_C_DEC**
- class **point_const_iterator_**

4.433.1 Detailed Description

Find type iterator.

Definition at line 43 of file point_iterator.hpp.

4.433.2 Member Typedef Documentation

4.433.2.1 typedef const_pointer_ point_iterator_::const_pointer

Iterator's const pointer type.

Definition at line 59 of file point_iterator.hpp.

4.433.2.2 typedef const_reference_ point_iterator_::const_reference

Iterator's const reference type.

Definition at line 65 of file point_iterator.hpp.

4.433.2.3 typedef trivial_iterator_difference_type point_iterator_::difference_type

Difference type.

Definition at line 50 of file point_iterator.hpp.

4.433.2.4 typedef trivial_iterator_tag point_iterator_::iterator_category

Category.

Definition at line 47 of file point_iterator.hpp.

4.433.2.5 typedef pointer_ point_iterator_::pointer

Iterator's pointer type.

Definition at line 56 of file point_iterator.hpp.

4.433.2.6 typedef reference_ point_iterator_::reference

Iterator's reference type.

Definition at line 62 of file point_iterator.hpp.

4.433.2.7 typedef value_type_ point_iterator_::value_type

Iterator's value type.

Definition at line 53 of file point_iterator.hpp.

4.433.3 Constructor & Destructor Documentation

4.433.3.1 point_iterator_::point_iterator_ ( ) `[inline]`

Default constructor.

Definition at line 69 of file point_iterator.hpp.

4.433.3.2 point_iterator_::point_iterator_ (const point_iterator_ & *other*) `[inline]`

Copy constructor.

Definition at line 75 of file point_iterator.hpp.

4.433.4 Member Function Documentation

4.433.4.1 bool point_iterator_::operator!= (const point_iterator_ & *other*) const `[inline]`

Compares content to a different iterator object.

Definition at line 107 of file point_iterator.hpp.

4.433.4.2 bool point_iterator_::operator!= (const point_const_iterator_ & *other*) const `[inline]`

Compares content (negatively) to a different iterator object.

Definition at line 112 of file point_iterator.hpp.

4.433.4.3 reference point_iterator_::operator* ( ) const [inline]

Access.

Definition at line 89 of file point_iterator.hpp.

4.433.4.4 pointer point_iterator_::operator-> ( ) const [inline]

Access.

Definition at line 81 of file point_iterator.hpp.

4.433.4.5 bool point_iterator_::operator== (const point_iterator_ & *other*) const [inline]

Compares content to a different iterator object.

Definition at line 97 of file point_iterator.hpp.

4.433.4.6 bool point_iterator_::operator== (const point_const_iterator_ & *other*) const [inline]

Compares content to a different iterator object.

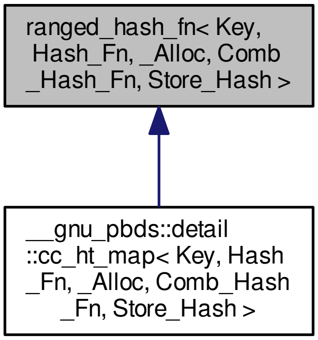
Definition at line 102 of file point_iterator.hpp.

The documentation for this class was generated from the following file:

- point_iterator.hpp

4.434 ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference

Inheritance diagram for ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >:



4.434.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>class ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >

Primary template.

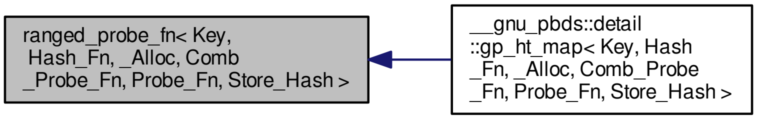
Definition at line 55 of file ranged_hash_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

4.435 ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference

Inheritance diagram for ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >:



4.435.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>class ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >

Primary template.

Definition at line 55 of file ranged_probe_fn.hpp.

The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

4.436 result_of< _Signature > Class Template Reference

4.436.1 Detailed Description

template<typename _Signature>class result_of< _Signature >

result_of

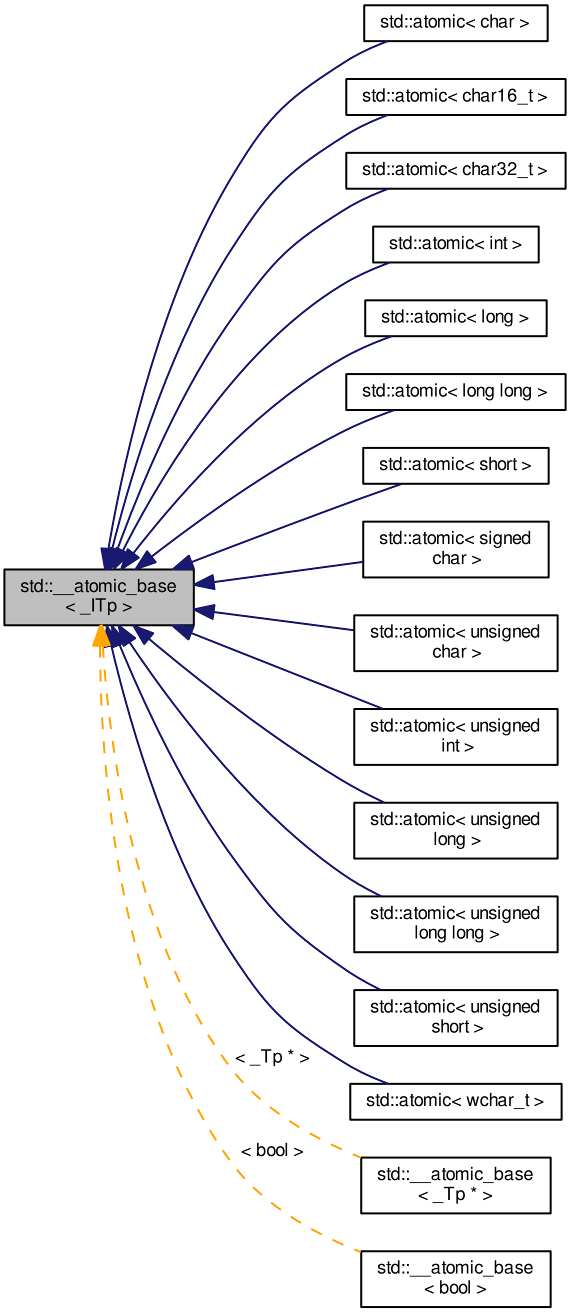
Definition at line 1878 of file type_traits.

The documentation for this class was generated from the following file:

- type_traits

4.437 std::__atomic_base< _ITp > Struct Template Reference

Inheritance diagram for std::__atomic_base< _ITp >:



Public Member Functions

- **__atomic_base** (const __atomic_base &)=delete
- constexpr **__atomic_base** (__int_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept

- __int_type **operator-=** (__int_type __i) volatilenoexcept
- __atomic_base & **operator=** (const __atomic_base &)=delete
- __atomic_base & **operator=** (const __atomic_base &) volatile=delete
- __int_type **operator=** (__int_type __i) noexcept
- __int_type **operator=** (__int_type __i) volatilenoexcept
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.437.1 Detailed Description

template<typename _ITp>struct std::__atomic_base< _ITp >

Base class for atomic integrals.

Definition at line 349 of file atomic_base.h.

The documentation for this struct was generated from the following file:

- atomic_base.h

4.438 std::__atomic_base< _PTp ∗ > Struct Template Reference

Public Member Functions

- **__atomic_base** (const __atomic_base &)=delete
- constexpr **__atomic_base** (__pointer_type __p) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatilenoexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __pointer_type** () const noexcept
- **operator __pointer_type** () const volatilenoexcept
- __pointer_type **operator++** (int) noexcept
- __pointer_type **operator++** (int) volatilenoexcept
- __pointer_type **operator++** () noexcept
- __pointer_type **operator++** () volatilenoexcept
- __pointer_type **operator+=** (ptrdiff_t __d) noexcept

- __pointer_type **operator+=** (ptrdiff_t __d) volatilenoexcept
- __pointer_type **operator--** (int) noexcept
- __pointer_type **operator--** (int) volatilenoexcept
- __pointer_type **operator--** () noexcept
- __pointer_type **operator--** () volatilenoexcept
- __pointer_type **operator-=** (ptrdiff_t __d) noexcept
- __pointer_type **operator-=** (ptrdiff_t __d) volatilenoexcept
- __atomic_base & **operator=** (const __atomic_base &)=delete
- __atomic_base & **operator=** (const __atomic_base &) volatile=delete
- __pointer_type **operator=** (__pointer_type __p) noexcept
- __pointer_type **operator=** (__pointer_type __p) volatilenoexcept
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.438.1 Detailed Description

template<typename _PTp>struct std::__atomic_base< _PTp * >

Partial specialization for pointer types.

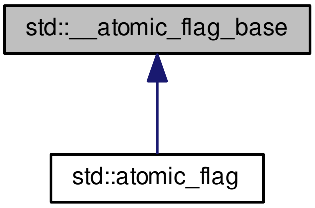
Definition at line 665 of file atomic_base.h.

The documentation for this struct was generated from the following file:

- atomic_base.h

4.439 std::__atomic_flag_base Struct Reference

Inheritance diagram for std::__atomic_flag_base:



Public Attributes

- __atomic_flag_data_type **_M_i**

4.439.1 Detailed Description

Base type for atomic_flag.

Base type is POD with data, allowing atomic_flag to derive from it and meet the standard layout type requirement. In addition to compatibilty with a C interface, this allows different implementations of atomic_flag to use the same atomic operation functions, via a standard conversion to the __atomic_flag_base argument.

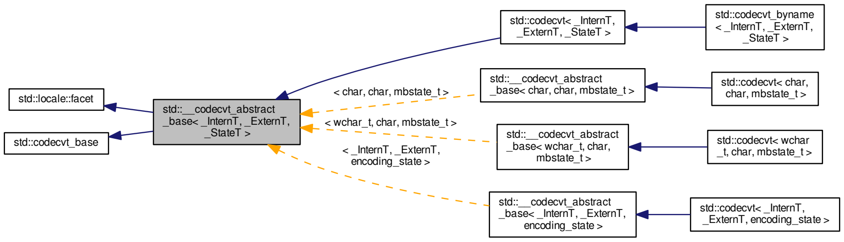
Definition at line 261 of file atomic_base.h.

The documentation for this struct was generated from the following file:

- atomic_base.h

4.440 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference

Inheritance diagram for std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- int **length** (state_type &__state, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- result unshift (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Protected Member Functions

- **__codecvt_abstract_base** (size_t __refs=0)
- virtual bool **do_always_noconv** () const =0 throw ()
- virtual int **do_encoding** () const =0 throw ()
- virtual result **do_in** (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const =0
- virtual int **do_length** (state_type &, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const =0
- virtual int **do_max_length** () const =0 throw ()
- virtual result do_out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const =0
- virtual result **do_unshift** (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.440.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 68 of file codecvt.h.

4.440.2 Member Function Documentation

4.440.2.1 template<typename _InternT, typename _ExternT, typename _StateT> virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out (state_type & *__state,* const intern_type ∗ *__from,* const intern_type ∗ *__from_end,* const intern_type ∗& *__from_next,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const `[protected]`, `[pure virtual]`

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See Also

out for more information.

Implemented in std::codecvt< wchar_t, char, mbstate_t >, std::codecvt< char, char, mbstate_t >, std::codecvt< _InternT, _ExternT, _StateT >, and std::codecvt< _InternT, _ExternT, encoding_state >.

Referenced by std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out().

4.440.2.2 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & *__state*, const extern_type * *__from*, const extern_type * *__from_end*, const extern_type *& *__from_next*, intern_type * *__to*, intern_type * *__to_end*, intern_type *& *__to_next*) const [inline]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.440.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (state_type & *__state*, const intern_type * *__from*, const intern_type * *__from_end*, const intern_type *& *__from_next*, extern_type * *__to*, extern_type * *__to_end*, extern_type *& *__to_next*) const [inline]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.440.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (state_type & *__state,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

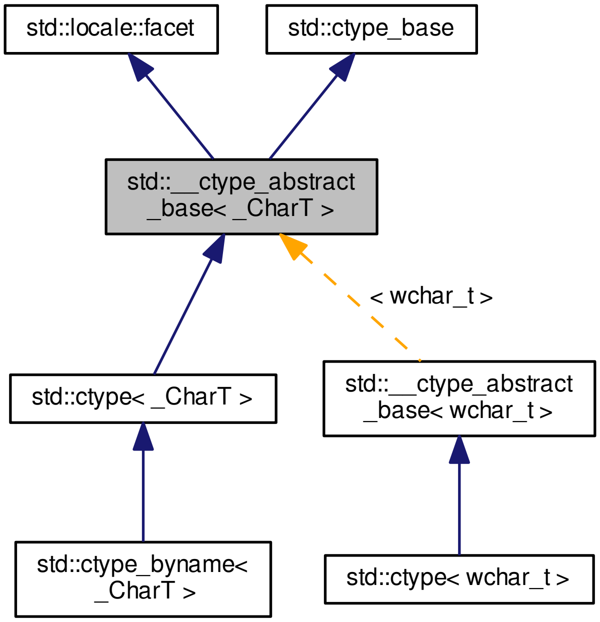
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt.h

4.441 std::__ctype_abstract_base< _CharT > Class Template Reference

Inheritance diagram for std::__ctype_abstract_base< _CharT >:



Public Types

- typedef const int ∗ **__to_type**
- typedef _CharT char_type
- typedef unsigned short **mask**

Public Member Functions

- bool is (mask __m, char_type __c) const
- const char_type ∗ is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- char narrow (char_type __c, char __dfault) const
- const char_type ∗ narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- const char_type ∗ scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- const char_type ∗ scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- char_type tolower (char_type __c) const
- const char_type ∗ tolower (char_type ∗__lo, const char_type ∗__hi) const
- char_type toupper (char_type __c) const
- const char_type ∗ toupper (char_type ∗__lo, const char_type ∗__hi) const
- char_type widen (char __c) const
- const char ∗ widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- **__ctype_abstract_base** (size_t __refs=0)
- virtual bool do_is (mask __m, char_type __c) const =0
- virtual const char_type ∗ do_is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const =0
- virtual char do_narrow (char_type __c, char __dfault) const =0
- virtual const char_type ∗ do_narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const =0
- virtual const char_type ∗ do_scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const =0
- virtual const char_type ∗ do_scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const =0
- virtual char_type do_tolower (char_type __c) const =0
- virtual const char_type ∗ do_tolower (char_type ∗__lo, const char_type ∗__hi) const =0
- virtual char_type do_toupper (char_type __c) const =0
- virtual const char_type ∗ do_toupper (char_type ∗__lo, const char_type ∗__hi) const =0
- virtual char_type do_widen (char __c) const =0
- virtual const char ∗ do_widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.441.1 Detailed Description

template<typename _CharT>class std::__ctype_abstract_base< _CharT >

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 143 of file locale_facets.h.

4.441.2 Member Typedef Documentation

4.441.2.1 template<typename _CharT> typedef _CharT **std::__ctype_abstract_base< _CharT >::char_type**

Typedef for the template parameter.

Definition at line 148 of file locale_facets.h.

4.441.3 Member Function Documentation

4.441.3.1 template<typename _CharT> virtual bool **std::__ctype_abstract_base< _CharT >::do_is** (mask *__m*, **char_type** *__c*) const [protected], [pure virtual]

Test char_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::is().

4.441.3.2 template<typename _CharT> virtual const **char_type*** **std::__ctype_abstract_base< _CharT >::do_is** (const **char_type** * *__lo*, const **char_type** * *__hi*, mask * *__vec*) const [protected], [pure virtual]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

4.441.3.3 template<typename _CharT> virtual char std::__ctype_abstract_base< _CharT >::do_narrow (char_type *__c*, char *__dfault*) const [protected], [pure virtual]

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::narrow().

4.441.3.4 template<typename _CharT> virtual const char_type∗ std::__ctype_abstract_base< _CharT >::do_narrow (const char_type ∗ *__lo*, const char_type ∗ *__hi*, char *__dfault*, char ∗ *__to*) const [protected], [pure virtual]

Narrow char_type array to char.

This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

4.441.3.5 template<typename _CharT> virtual const char_type∗ std::__ctype_abstract_base< _CharT >::do_scan_is (mask __m, const char_type ∗ __lo, const char_type ∗ __hi) const [protected], [pure virtual]

Find char_type matching mask.

This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::scan_is().

4.441.3.6 template<typename _CharT> virtual const char_type∗ std::__ctype_abstract_base< _CharT >::do_scan_not (mask __m, const char_type ∗ __lo, const char_type ∗ __hi) const [protected], [pure virtual]

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::scan_not().

4.441.3.7 template<typename _CharT> virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (char_type __c) const [protected], [pure virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::tolower().

4.441.3.8 template<typename _CharT> virtual const **char_type**∗ **std::__ctype_abstract_base**< _CharT >::do_tolower (**char_type** ∗ *__lo,* const **char_type** ∗ *__hi*) const [protected], [pure virtual]

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

4.441.3.9 template<typename _CharT> virtual **char_type std::__ctype_abstract_base**< _CharT >::do_toupper (**char_type** *__c*) const [protected], [pure virtual]

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::toupper().

4.441.3.10 template<typename _CharT> virtual const char_type* std::__ctype_abstract_base< _CharT >::do_toupper (char_type * *__lo,* const char_type * *__hi*) const [protected], [pure virtual]

Convert array to uppercase.

This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

4.441.3.11 template<typename _CharT> virtual char_type std::__ctype_abstract_base< _CharT >::do_widen (char *__c*) const [protected], [pure virtual]

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

Referenced by std::__ctype_abstract_base< wchar_t >::widen().

4.441.3.12 template<typename _CharT> virtual const char* std::__ctype_abstract_base< _CharT >::do_widen (const char * *__lo,* const char * *__hi,* char_type * *__to*) const [protected], [pure virtual]

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< wchar_t >, and std::ctype< _CharT >.

4.441.3.13 template<typename _CharT> bool **std::__ctype_abstract_base< _CharT >::is (mask *__m*, char_type *__c*) const** `[inline]`

Test char_type classification.

This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__c	The char_type to compare the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Definition at line 162 of file locale_facets.h.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.441.3.14 template<typename _CharT> const char_type∗ **std::__ctype_abstract_base< _CharT >::is (const char_type ∗ *__lo*, const char_type ∗ *__hi*, mask ∗ *__vec*) const** `[inline]`

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Definition at line 179 of file locale_facets.h.

4.441.3.15 template<typename _CharT> char **std::__ctype_abstract_base< _CharT >::narrow (char_type *__c*, char *__dfault*) const** `[inline]`

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file locale_facets.h.

Referenced by std::time_put< _CharT, _OutIter >::put().

4.441.3.16 template<typename _CharT> const **char_type**∗ **std::__ctype_abstract_base**< _CharT >::narrow (const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi,* char *__dfault,* char ∗ *__to*) const [inline]

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 346 of file locale_facets.h.

4.441.3.17 template<typename _CharT> const **char_type**∗ **std::__ctype_abstract_base**< _CharT >::scan_is (mask *__m,* const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi*) const [inline]

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __*hi*.

Definition at line 195 of file locale_facets.h.

4.441.3.18 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::scan_not (mask __*m*, const char_type * __*lo*, const char_type * __*hi*) const [inline]

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

__*m*	The mask to compare against.
__*lo*	Pointer to first char in range.
__*hi*	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

Definition at line 211 of file locale_facets.h.

4.441.3.19 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::tolower (char_type __*c*) const [inline]

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

__*c*	The char_type to convert.

Returns

The lowercase char_type if convertible, else __*c*.

Definition at line 254 of file locale_facets.h.

4.441.3.20 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::tolower (char_type * __*lo*, const char_type * __*hi*) const [inline]

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

__*lo*	Pointer to start of range.
__*hi*	Pointer to end of range.

Returns

__hi.

Definition at line 269 of file locale_facets.h.

4.441.3.21 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::toupper (char_type *__c*) const [inline]

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Definition at line 225 of file locale_facets.h.

4.441.3.22 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::toupper (char_type * *__lo*, const char_type * *__hi*) const [inline]

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 240 of file locale_facets.h.

4.441.3.23 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::widen (char *__c*) const [inline]

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_put< _CharT, _OutIter >::do_put(), std::money_put< _CharT, _OutIter >::do_put(), std::tr2::operator<<(), and std::operator<<().

4.441.3.24 template<typename _CharT> const char* **std::__ctype_abstract_base< _CharT >::widen** (const char * *__lo,* const char * *__hi,* **char_type** * *__to*) const `[inline]`

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 305 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.442 std::__debug::bitset< _Nb > Class Template Reference

Inherits bitset< _Nb >.

Public Types

- typedef _Base::reference **reference**

Public Member Functions

- constexpr **bitset** (unsigned long long __val) noexcept
- template<typename _CharT , typename _Traits , typename _Alloc >
  bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __pos=0, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n=(std::basic_string< _CharT, _Traits, _Alloc >::npos))
- template<class _CharT , class _Traits , class _Alloc >
  bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __pos, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))
- **bitset** (const _Base &__x)

- template<typename _CharT >
  bitset (const _CharT ∗__str, typename std::basic_string< _CharT >::size_type __n=std::basic_string< _CharT >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- bitset< _Nb > & **flip** () noexcept
- bitset< _Nb > & **flip** (size_t __pos)
- bool **operator!=** (const bitset< _Nb > &__rhs) const noexcept
- bitset< _Nb > & **operator&=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > **operator<<** (size_t __pos) const noexcept
- bitset< _Nb > & **operator<<=** (size_t __pos) noexcept
- bool **operator==** (const bitset< _Nb > &__rhs) const noexcept
- bitset< _Nb > **operator>>** (size_t __pos) const noexcept
- bitset< _Nb > & **operator>>=** (size_t __pos) noexcept
- reference **operator[ ]** (size_t __pos)
- constexpr bool **operator[ ]** (size_t __pos) const
- bitset< _Nb > & **operator^=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > & **operator|=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > **operator~** () const noexcept
- bitset< _Nb > & **reset** () noexcept
- bitset< _Nb > & **reset** (size_t __pos)
- bitset< _Nb > & **set** () noexcept
- bitset< _Nb > & **set** (size_t __pos, bool __val=true)
- template<typename _CharT , typename _Traits , typename _Alloc >
  std::basic_string< _CharT,
  _Traits, _Alloc > **to_string** () const
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_string< _CharT,
  _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<typename _CharT , typename _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **to_string** () const
- template<class _CharT , class _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<typename _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **to_string** () const
- template<class _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- std::basic_string< char,
  std::char_traits< char >
  , std::allocator< char > > **to_string** () const
- std::basic_string< char,
  std::char_traits< char >
  , std::allocator< char > > **to_string** (char __zero, char __one= '1') const

4.442.1 Detailed Description

template<size_t _Nb>class std::__debug::bitset< _Nb >

Class std::bitset with additional safety/checking/debug instrumentation.

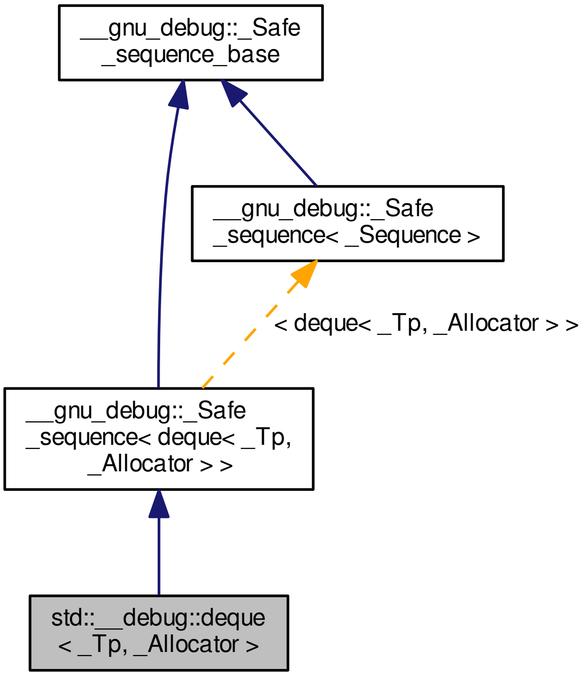
Definition at line 42 of file debug/bitset.

The documentation for this class was generated from the following file:

- debug/bitset

4.443 std::__debug::deque< _Tp, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::deque< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, deque > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**

- typedef std::reverse_iterator
  < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef
  __gnu_debug::_Safe_iterator
  < _Base_iterator, deque > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **deque** (const _Allocator &__a=_Allocator())
- **deque** (size_type __n)
- **deque** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  deque (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **deque** (const deque &__x)
- **deque** (const _Base &__x)
- **deque** (deque &&__x)
- **deque** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- void **assign** (initializer_list< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (iterator __position, _Args &&...__args)
- template<typename... _Args>
  void **emplace_back** (_Args &&...__args)

- template<typename... _Args>
  void **emplace_front** (_Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, initializer_list< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- deque & **operator=** (const deque &__x)
- deque & **operator=** (deque &&__x)
- deque & **operator=** (initializer_list< value_type > __l)
- reference **operator[ ]** (size_type __n)
- const_reference **operator[ ]** (size_type __n) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (const _Tp &__x)
- void **push_back** (_Tp &&__x)
- void **push_front** (const _Tp &__x)
- void **push_front** (_Tp &&__x)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **shrink_to_fit** ()
- void **swap** (deque &__x)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.443.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__debug::deque< _Tp, _Allocator >

Class std::deque with safety/checking/debug instrumentation.

Definition at line 42 of file debug/deque.

4.443.2 Member Function Documentation

4.443.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]

Attach an iterator to this sequence.

4.443.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.443.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.443.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.443.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw () [inherited]

Likewise but not thread safe.

4.443.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.443.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.443.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.443.2.9 void **__gnu_debug::_Safe_sequence**< **deque**< _Tp, _Allocator > >::_M_invalidate_if (_Predicate *__pred*) `[inherited]`

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.443.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.443.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (**_Safe_sequence_base** & *__x*) `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.443.2.12 void **__gnu_debug::_Safe_sequence**< **deque**< _Tp, _Allocator > >::_M_transfer_from_if (**_Safe_sequence**< **deque**< _Tp, _Allocator > > & *__from*, _Predicate *__pred*) `[inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.443.3 Member Data Documentation

4.443.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.443.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.443.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

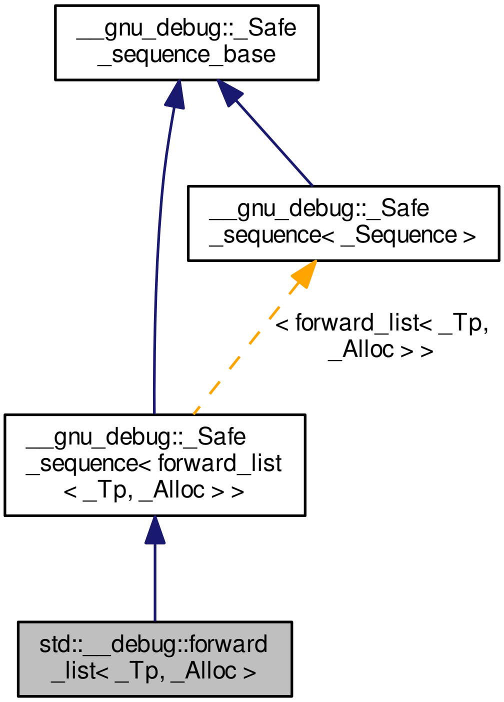
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/deque

4.444 std::__debug::forward_list< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::__debug::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, forward_list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, forward_list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **forward_list** (const _Alloc &__al=_Alloc())
- **forward_list** (const forward_list &__list, const _Alloc &__al)
- **forward_list** (forward_list &&__list, const _Alloc &__al)
- **forward_list** (size_type __n, const _Alloc &__al=_Alloc())
- **forward_list** (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- **forward_list** (const forward_list &__list)
- **forward_list** (forward_list &&__list) noexcept
- **forward_list** (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__val)
- void **assign** (std::initializer_list< _Tp > __il)
- iterator **before_begin** () noexcept
- const_iterator **before_begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbefore_begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace_after** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase_after** (const_iterator __pos)
- iterator **erase_after** (const_iterator __pos, const_iterator __last)
- reference **front** ()
- const_reference **front** () const
- iterator **insert_after** (const_iterator __pos, const _Tp &__val)
- iterator **insert_after** (const_iterator __pos, _Tp &&__val)
- iterator **insert_after** (const_iterator __pos, size_type __n, const _Tp &__val)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert_after** (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- iterator **insert_after** (const_iterator __pos, std::initializer_list< _Tp > __il)
- void **merge** (forward_list &&__list)
- void **merge** (forward_list &__list)
- template<typename _Comp >
  void **merge** (forward_list &&__list, _Comp __comp)

- template<typename _Comp >
  void **merge** (forward_list &__list, _Comp __comp)
- forward_list & **operator=** (const forward_list &__list)
- forward_list & **operator=** (forward_list &&__list) noexcept(_Node_alloc_traits
- forward_list & **operator=** (std::initializer_list< _Tp > __il)
- void **pop_front** ()
- void **remove** (const _Tp &__val)
- template<typename _Pred >
  void **remove_if** (_Pred __pred)
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const value_type &__val)
- void **splice_after** (const_iterator __pos, forward_list &&__list)
- void **splice_after** (const_iterator __pos, forward_list &__list)
- void **splice_after** (const_iterator __pos, forward_list &&__list, const_iterator __i)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __i)
- void **splice_after** (const_iterator __pos, forward_list &&__list, const_iterator __before, const_iterator __last)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __before, const_iterator __last)
- void **swap** (forward_list &__list) noexcept(_Node_alloc_traits
- void **unique** ()
- template<typename _BinPred >
  void **unique** (_BinPred __binary_pred)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.444.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>class std::__debug::forward_list< _Tp, _Alloc >

Class std::forward_list with safety/checking/debug instrumentation.

Definition at line 44 of file debug/forward_list.

4.444.2 Member Function Documentation

4.444.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ *__it,* bool *__constant*) [inherited]

Attach an iterator to this sequence.

4.444.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.444.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ __it) [inherited]

Detach an iterator from this sequence

4.444.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.444.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.444.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.444.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.444.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.444.2.9 void __gnu_debug::_Safe_sequence< forward_list< _Tp, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.444.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.444.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.444.2.12 void __gnu_debug::_Safe_sequence< forward_list< _Tp, _Alloc > >::_M_transfer_from_if (_Safe_sequence< forward_list< _Tp, _Alloc > > & __from, _Predicate __pred) [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.444.3 Member Data Documentation

4.444.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.444.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.444.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

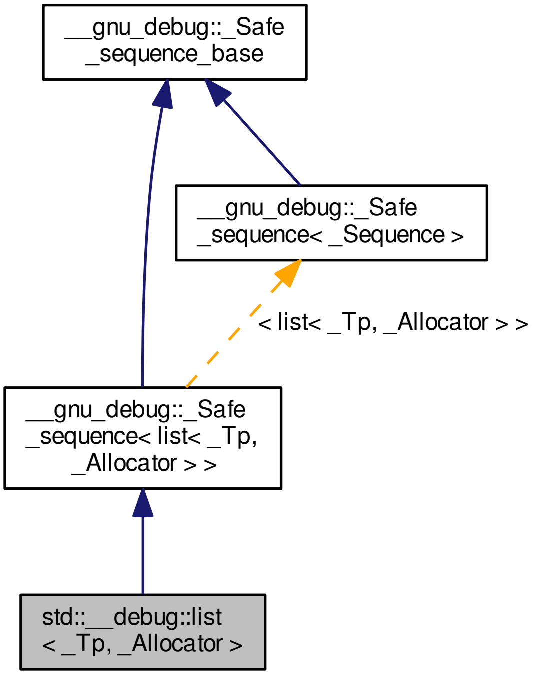
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/forward_list

4.445 std::__debug::list< _Tp, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::list< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **list** (const _Allocator &__a=_Allocator())
- **list** (size_type __n)
- **list** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** (const list &__x)
- **list** (const _Base &__x)
- **list** (list &&__x) noexcept
- **list** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base *__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base *__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base *__it)
- void _M_detach_single (_Safe_iterator_base *__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- void **assign** (initializer_list< value_type > __l)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (iterator __position, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __position, iterator __last)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, initializer_list< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- void **merge** (list &&__x)
- void **merge** (list &__x)
- template<class _Compare >
  void **merge** (list &&__x, _Compare __comp)

- template<typename _Compare >
  void **merge** (list &__x, _Compare __comp)
- list & **operator=** (const list &__x)
- list & **operator=** (list &&__x)
- list & **operator=** (initializer_list< value_type > __l)
- void **pop_back** ()
- void **pop_front** ()
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- void **remove** (const _Tp &__value)
- template<class _Predicate >
  void **remove_if** (_Predicate __pred)
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **sort** ()
- template<typename _StrictWeakOrdering >
  void **sort** (_StrictWeakOrdering __pred)
- void **splice** (iterator __position, list &&__x)
- void **splice** (iterator __position, list &__x)
- void **splice** (iterator __position, list &&__x, iterator __i)
- void **splice** (iterator __position, list &__x, iterator __i)
- void **splice** (iterator __position, list &&__x, iterator __first, iterator __last)
- void **splice** (iterator __position, list &__x, iterator __first, iterator __last)
- void **swap** (list &__x)
- void **unique** ()
- template<class _BinaryPredicate >
  void **unique** (_BinaryPredicate __binary_pred)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.445.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__debug::list< _Tp, _Allocator >

Class std::list with safety/checking/debug instrumentation.

Definition at line 42 of file debug/list.

4.445.2 Member Function Documentation

4.445.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]

Attach an iterator to this sequence.

4.445.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.445.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * __it) [inherited]

Detach an iterator from this sequence

4.445.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::~_Safe_sequence_base().

4.445.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * __it) throw () [inherited]

Likewise but not thread safe.

4.445.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.445.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.445.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.445.2.9 void __gnu_debug::_Safe_sequence< list< _Tp, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which __`pred(x)` returns `true`. __`pred` will be invoked with the normal iterators nested in the safe ones.

4.445.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.445.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (**_Safe_sequence_base & __x**) `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.445.2.12 void **__gnu_debug::_Safe_sequence**< **list**< _Tp, _Allocator > >::_M_transfer_from_if (**_Safe_sequence**< **list**< _Tp, _Allocator > > & *__from,* _Predicate *__pred*) `[inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which __`pred(x)` returns `true`. __`pred` will be invoked with the normal iterators nested in the safe ones.

4.445.3 Member Data Documentation

4.445.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.445.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.445.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

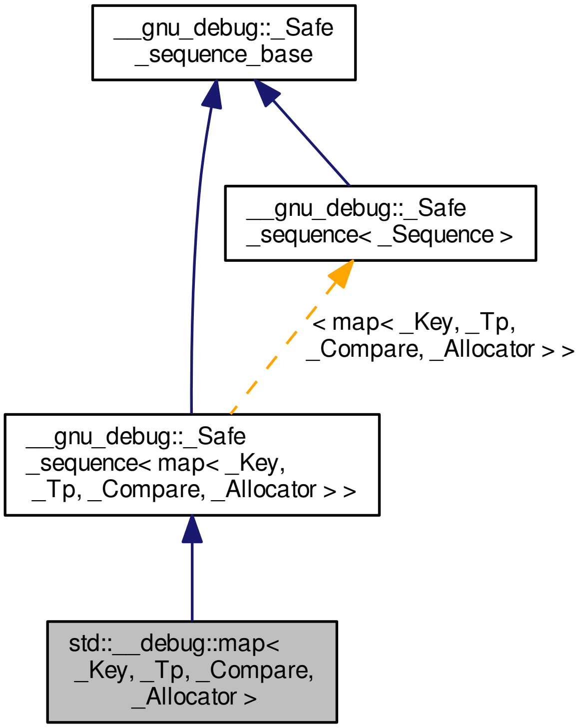
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/list

4.446 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::map< _Key, _Tp, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, map > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, map > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**

- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key,
  _Tp > **value_type**

Public Member Functions

- **map** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **map** (const map &__x)
- **map** (const _Base &__x)
- **map** (map &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **map** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- std::pair< iterator, bool > **insert** (const value_type &__x)

- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__x)
- void **insert** (std::initializer_list< value_type > __list)
- iterator **insert** (const_iterator __position, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- map & **operator=** (const map &__x)
- map & **operator=** (map &&__x)
- map & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (map &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.446.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>class std::__debug::map< _Key, _Tp, _Compare, _Allocator >

Class std::map with safety/checking/debug instrumentation.

Definition at line 43 of file debug/map.h.

4.446.2 Member Function Documentation

4.446.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ *__it,* bool *__constant*) `[inherited]`

Attach an iterator to this sequence.

4.446.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.446.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ __it) [inherited]

Detach an iterator from this sequence

4.446.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.446.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.446.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.446.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.446.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.446.2.9 void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.446.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.446.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.446.2.12 void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if (_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > > & __from, _Predicate __pred) `[inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.446.3 Member Data Documentation

4.446.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.446.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.446.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

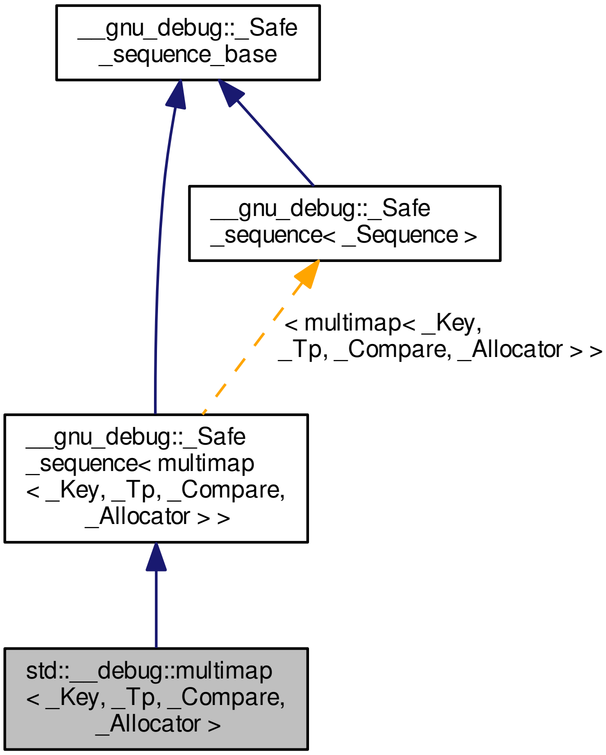
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/map.h

4.447 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, multimap > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, multimap > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**

- typedef _Tp **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key,
  _Tp > **value_type**

Public Member Functions

- **multimap** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
  multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multimap** (const multimap &__x)
- **multimap** (const _Base &__x)
- **multimap** (multimap &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **multimap** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const

- iterator **insert** (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__x)
- void **insert** (std::initializer_list< value_type > __list)
- iterator **insert** (const_iterator __position, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multimap & **operator=** (const multimap &__x)
- multimap & **operator=** (multimap &&__x)
- multimap & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (multimap &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.447.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >

Class std::multimap with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multimap.h.

4.447.2 Member Function Documentation

4.447.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (**_Safe_iterator_base** ∗ *__it,* bool *__constant*) [inherited]

Attach an iterator to this sequence.

4.447.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.447.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ __it) [inherited]

Detach an iterator from this sequence

4.447.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.447.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.447.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.447.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.447.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.447.2.9 void __gnu_debug::_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.447.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.447.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.447.2.12 void __gnu_debug::_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if (_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > > & *__from,* _Predicate *__pred*) `[inherited]`

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.447.3 Member Data Documentation

4.447.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.447.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.447.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

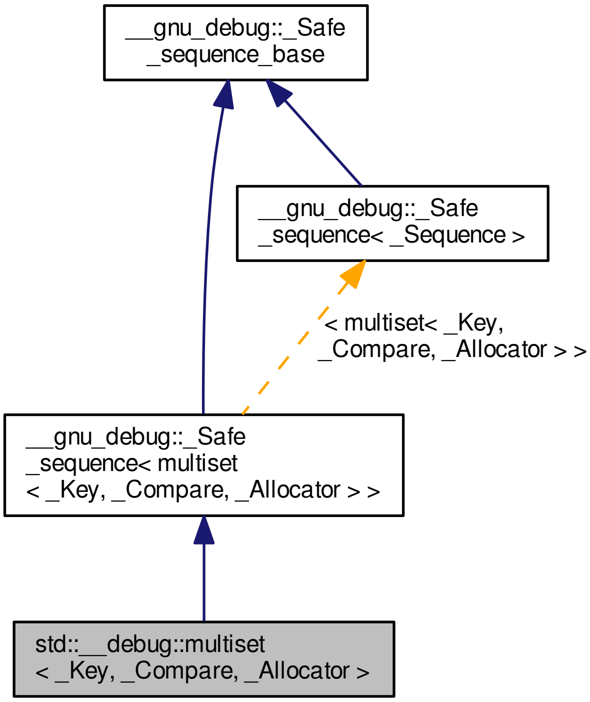
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/multimap.h

4.448 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::multiset< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, multiset > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, multiset > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**

- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator >
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multiset** (const multiset &__x)
- **multiset** (const _Base &__x)
- **multiset** (multiset &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **multiset** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base *__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base *__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base *__it)
- void _M_detach_single (_Safe_iterator_base *__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const value_type &__x)
- iterator **insert** (value_type &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)

- iterator **insert** (const_iterator __position, value_type &&__x)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list< value_type > __l)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multiset & **operator=** (const multiset &__x)
- multiset & **operator=** (multiset &&__x)
- multiset & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (multiset &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.448.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>class std::__debug::multiset< _Key, _Compare, _Allocator >

Class std::multiset with safety/checking/debug instrumentation.

Definition at line 43 of file debug/multiset.h.

4.448.2 Member Function Documentation

4.448.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ *__it,* bool *__constant*) [inherited]

Attach an iterator to this sequence.

4.448.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.448.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ __it) [inherited]

Detach an iterator from this sequence

4.448.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.448.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.448.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.448.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.448.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.448.2.9 void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.448.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.448.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.448.2.12 void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_transfer_from_if (_Safe_sequence< multiset< _Key, _Compare, _Allocator > > & __from, _Predicate __pred) [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.448.3 Member Data Documentation

4.448.3.1 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.448.3.2 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.448.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

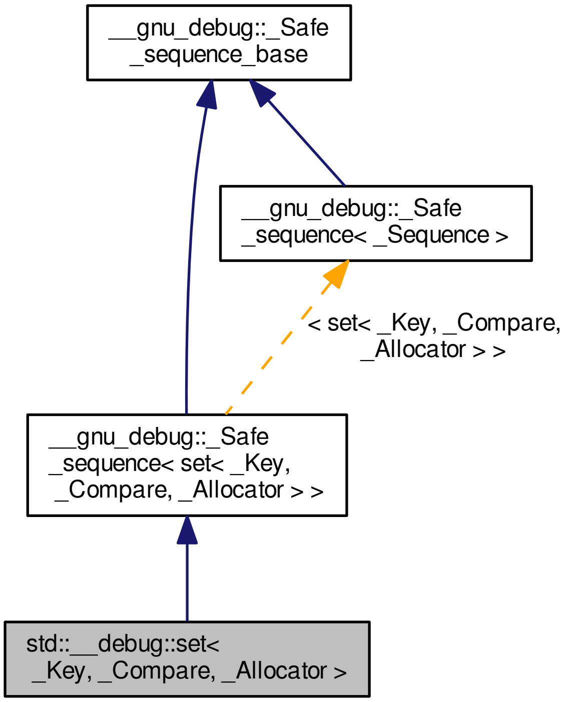
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/multiset.h

4.449 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::set< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, set > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, set > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**

- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **set** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator > **set** (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **set** (const set &__x)
- **set** (const _Base &__x)
- **set** (set &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **set** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args> std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args> iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- std::pair< iterator, bool > **insert** (const value_type &__x)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)

- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list< value_type > __l)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- set & **operator=** (const set &__x)
- set & **operator=** (set &&__x)
- set & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (set &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.449.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>class std::__debug::set< _Key, _Compare, _Allocator >

Class std::set with safety/checking/debug instrumentation.

Definition at line 43 of file debug/set.h.

4.449.2 Member Function Documentation

4.449.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ *__it,* bool *__constant*) [inherited]

Attach an iterator to this sequence.

4.449.2.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.449.2.3 void __gnu_debug::_Safe_sequence_base::_M_detach (**_Safe_iterator_base** * __it) [inherited]

Detach an iterator from this sequence

4.449.2.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.449.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (**_Safe_iterator_base** * __it) throw () [inherited]

Likewise but not thread safe.

4.449.2.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

> for all iterators i attached to this sequence, i->_M_version == _M_version.

4.449.2.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.449.2.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.449.2.9 void **__gnu_debug::_Safe_sequence**< **set**< _Key, _Compare, _Allocator > >::_M_invalidate_if (_Predicate *__pred*) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.449.2.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.449.2.11 void __gnu_debug::_Safe_sequence_base::_M_swap (**_Safe_sequence_base** & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.449.2.12 void **__gnu_debug::_Safe_sequence**< **set**< _Key, _Compare, _Allocator > >::_M_transfer_from_if (**_Safe_sequence**< **set**< _Key, _Compare, _Allocator > > & *__from*, _Predicate *__pred*) [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.449.3 Member Data Documentation

4.449.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.449.3.2 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.449.3.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version `[mutable]`, `[inherited]`

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

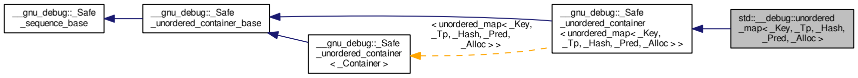
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/set.h

4.450 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Inheritance diagram for std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, unordered_map > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator < _Base_const_local_iterator, unordered_map > **const_local_iterator**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, unordered_map > **iterator**

- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef
  __gnu_debug::_Safe_local_iterator
  < _Base_local_iterator,
  unordered_map > **local_iterator**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const unordered_map &__x)=default
- **unordered_map** (const _Base &__x)
- **unordered_map** (unordered_map &&__x)=default
- **unordered_map** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base *__it, bool __constant)
- void _M_attach_local (_Safe_iterator_base *__it, bool __constant)
- void _M_attach_local_single (_Safe_iterator_base *__it, bool __constant) throw ()
- void _M_attach_single (_Safe_iterator_base *__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base *__it)
- void _M_detach_local (_Safe_iterator_base *__it)
- void _M_detach_local_single (_Safe_iterator_base *__it) throw ()
- void _M_detach_single (_Safe_iterator_base *__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_invalidate_local_if (_Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- local_iterator **end** (size_type __b)

- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__obj)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __hint, _Pair &&__obj)
- void **insert** (std::initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_map & **operator=** (const unordered_map &__x)
- unordered_map & **operator=** (unordered_map &&__x)
- unordered_map & **operator=** (initializer_list< value_type > __l)
- void **swap** (unordered_map &__x)

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_unordered_container_base &__x)
- void _M_swap (_Safe_sequence_base &__x)

4.450.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_map with safety/checking/debug instrumentation.

Definition at line 50 of file debug/unordered_map.

4.450.2 Member Function Documentation

4.450.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ *__it,* bool *__constant*) [inherited]

Attach an iterator to this sequence.

4.450.2.2 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (_Safe_iterator_base ∗ *__it,* bool *__constant*) [inherited]

Attach an iterator to this container.

4.450.2.3 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.450.2.4 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.450.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ *__it*) [inherited]

Detach an iterator from this sequence

4.450.2.6 void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

4.450.2.7 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (_Safe_iterator_base ∗ *__it*) [inherited]

Detach an iterator from this container

4.450.2.8 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (_Safe_iterator_base ∗ *__it*) throw () [inherited]

Likewise but not thread safe.

4.450.2.9 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ *__it*) throw () [inherited]

Likewise but not thread safe.

4.450.2.10 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.450.2.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.450.2.12 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.450.2.13 void __gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.450.2.14 void __gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_invalidate_local_if (_Predicate __pred) [inherited]

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.450.2.15 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.450.2.16 void __gnu_debug::_Safe_unordered_container_base::_M_swap (_Safe_unordered_container_base & __x) [protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.450.2.17 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.450.3 Member Data Documentation

4.450.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.450.3.2 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.450.3.3 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.450.3.4 _Safe_iterator_base∗ __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.450.3.5 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

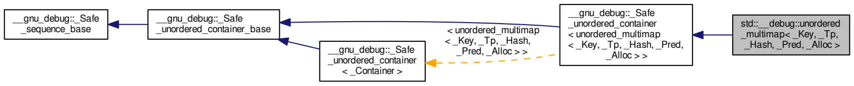
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/unordered_map

4.451 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Inheritance diagram for std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, unordered_multimap > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator < _Base_const_local_iterator, unordered_multimap > **const_local_iterator**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, unordered_multimap > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator < _Base_local_iterator, unordered_multimap > **local_iterator**
- typedef _Base::size_type **size_type**

- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multimap** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (const unordered_multimap &__x)=default
- **unordered_multimap** (const _Base &__x)
- **unordered_multimap** (unordered_multimap &&__x)=default
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_local (_Safe_iterator_base ∗__it)
- void _M_detach_local_single (_Safe_iterator_base ∗__it) throw ()
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_invalidate_local_if (_Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)

- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__obj)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __hint, _Pair &&__obj)
- void **insert** (std::initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_multimap & **operator=** (const unordered_multimap &__x)
- unordered_multimap & **operator=** (unordered_multimap &&__x)
- unordered_multimap & **operator=** (initializer_list< value_type > __l)
- void **swap** (unordered_multimap &__x)

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_unordered_container_base &__x)
- void _M_swap (_Safe_sequence_base &__x)

4.451.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_multimap with safety/checking/debug instrumentation.

Definition at line 478 of file debug/unordered_map.

4.451.2 Member Function Documentation

4.451.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]

Attach an iterator to this sequence.

4.451.2.2 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (_Safe_iterator_base ∗ __it, bool __constant) [inherited]

Attach an iterator to this container.

4.451.2.3 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (_Safe_iterator_base ∗ __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.451.2.4 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ __it, bool __constant) throw () [inherited]

Likewise but not thread safe.

4.451.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ __it) [inherited]

Detach an iterator from this sequence

4.451.2.6 void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

4.451.2.7 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (_Safe_iterator_base ∗ __it) [inherited]

Detach an iterator from this container

4.451.2.8 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.451.2.9 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ __it) throw () [inherited]

Likewise but not thread safe.

4.451.2.10 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.451.2.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.451.2.12 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.451.2.13 void __gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.451.2.14 void __gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > >::_M_invalidate_local_if (_Predicate __pred) [inherited]

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.451.2.15 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.451.2.16 void __gnu_debug::_Safe_unordered_container_base::_M_swap (_Safe_unordered_container_base & __x) [protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.451.2.17 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.451.3 Member Data Documentation

4.451.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.451.3.2 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.451.3.3 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.451.3.4 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.451.3.5 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

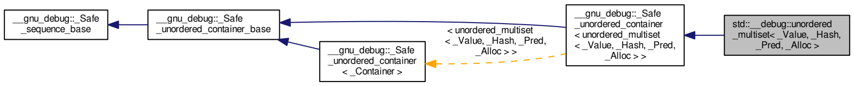
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/unordered_map

4.452 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

Inheritance diagram for std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, unordered_multiset > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator < _Base_const_local_iterator, unordered_multiset > **const_local_iterator**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, unordered_multiset > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator < _Base_local_iterator, unordered_multiset > **local_iterator**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())

- template<typename _InputIterator >
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const unordered_multiset &__x)=default
- **unordered_multiset** (const _Base &__x)
- **unordered_multiset** (unordered_multiset &&__x)=default
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_local_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_local (_Safe_iterator_base ∗__it)
- void _M_detach_local_single (_Safe_iterator_base ∗__it) throw ()
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_invalidate_local_if (_Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (value_type &&__obj)

- iterator **insert** (const_iterator __hint, value_type &&__obj)
- void **insert** (std::initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_multiset & **operator=** (const unordered_multiset &__x)
- unordered_multiset & **operator=** (unordered_multiset &&__x)
- unordered_multiset & **operator=** (initializer_list< value_type > __l)
- void **swap** (unordered_multiset &__x)

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_const_local_iterators
- _Safe_iterator_base ∗ _M_iterators
- _Safe_iterator_base ∗ _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_unordered_container_base &__x)
- void _M_swap (_Safe_sequence_base &__x)

4.452.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

Class std::unordered_multiset with safety/checking/debug instrumentation.

Definition at line 474 of file debug/unordered_set.

4.452.2 Member Function Documentation

4.452.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base ∗ __it, bool __constant) [inherited]

Attach an iterator to this sequence.

4.452.2.2 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (_Safe_iterator_base ∗ __it, bool __constant) [inherited]

Attach an iterator to this container.

4.452.2.3 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.452.2.4 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base ∗ *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.452.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base ∗ *__it*) [inherited]

Detach an iterator from this sequence

4.452.2.6 void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

4.452.2.7 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (_Safe_iterator_base ∗ *__it*) [inherited]

Detach an iterator from this container

4.452.2.8 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (_Safe_iterator_base ∗ *__it*) throw () [inherited]

Likewise but not thread safe.

4.452.2.9 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base ∗ *__it*) throw () [inherited]

Likewise but not thread safe.

4.452.2.10 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.452.2.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.452.2.12 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.452.2.13 void __gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate *__pred*) `[inherited]`

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.452.2.14 void __gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_local_if (_Predicate *__pred*) `[inherited]`

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.452.2.15 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) `[protected]`, `[inherited]`

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.452.2.16 void __gnu_debug::_Safe_unordered_container_base::_M_swap (_Safe_unordered_container_base & *__x*) `[protected]`, `[inherited]`

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.452.2.17 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & *__x*) `[protected]`, `[inherited]`

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.452.3 Member Data Documentation

4.452.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators `[inherited]`

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.452.3.2 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators `[inherited]`

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.452.3.3 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators `[inherited]`

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.452.3.4 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators `[inherited]`

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.452.3.5 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

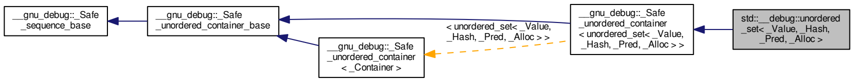
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/unordered_set

4.453 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

Inheritance diagram for std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator < _Base_const_iterator, unordered_set > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator < _Base_const_local_iterator, unordered_set > **const_local_iterator**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator < _Base_iterator, unordered_set > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator < _Base_local_iterator, unordered_set > **local_iterator**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())

- template<typename _InputIterator >
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (const unordered_set &__x)=default
- **unordered_set** (const _Base &__x)
- **unordered_set** (unordered_set &&__x)=default
- **unordered_set** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base *__it, bool __constant)
- void _M_attach_local (_Safe_iterator_base *__it, bool __constant)
- void _M_attach_local_single (_Safe_iterator_base *__it, bool __constant) throw ()
- void _M_attach_single (_Safe_iterator_base *__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base *__it)
- void _M_detach_local (_Safe_iterator_base *__it)
- void _M_detach_local_single (_Safe_iterator_base *__it) throw ()
- void _M_detach_single (_Safe_iterator_base *__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_invalidate_local_if (_Predicate __pred)
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- std::pair< iterator, bool > **insert** (value_type &&__obj)

- iterator **insert** (const_iterator __hint, value_type &&__obj)
- void **insert** (std::initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_set & **operator=** (const unordered_set &__x)
- unordered_set & **operator=** (unordered_set &&__x)
- unordered_set & **operator=** (initializer_list< value_type > __l)
- void **swap** (unordered_set &__x)

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_unordered_container_base &__x)
- void _M_swap (_Safe_sequence_base &__x)

4.453.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >

Class std::unordered_set with safety/checking/debug instrumentation.

Definition at line 50 of file debug/unordered_set.

4.453.2 Member Function Documentation

4.453.2.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * __it, bool __constant) [inherited]

Attach an iterator to this sequence.

4.453.2.2 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local (_Safe_iterator_base * __it, bool __constant) [inherited]

Attach an iterator to this container.

4.453.2.3 void __gnu_debug::_Safe_unordered_container_base::_M_attach_local_single (**_Safe_iterator_base** ∗ *__it,* bool *__constant*) throw () `[inherited]`

Likewise but not thread safe.

4.453.2.4 void __gnu_debug::_Safe_sequence_base::_M_attach_single (**_Safe_iterator_base** ∗ *__it,* bool *__constant*) throw () `[inherited]`

Likewise but not thread safe.

4.453.2.5 void __gnu_debug::_Safe_sequence_base::_M_detach (**_Safe_iterator_base** ∗ *__it*) `[inherited]`

Detach an iterator from this sequence

4.453.2.6 void __gnu_debug::_Safe_unordered_container_base::_M_detach_all ( ) `[protected]`, `[inherited]`

Detach all iterators, leaving them singular.

4.453.2.7 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local (**_Safe_iterator_base** ∗ *__it*) `[inherited]`

Detach an iterator from this container

4.453.2.8 void __gnu_debug::_Safe_unordered_container_base::_M_detach_local_single (**_Safe_iterator_base** ∗ *__it*) throw () `[inherited]`

Likewise but not thread safe.

4.453.2.9 void __gnu_debug::_Safe_sequence_base::_M_detach_single (**_Safe_iterator_base** ∗ *__it*) throw () `[inherited]`

Likewise but not thread safe.

4.453.2.10 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) `[protected]`, `[inherited]`

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.453.2.11 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () `[protected]`, `[inherited]`

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.453.2.12 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const `[inline]`, `[inherited]`

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.453.2.13 void __gnu_debug::_Safe_unordered_container< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.453.2.14 void __gnu_debug::_Safe_unordered_container< unordered_set< _Value, _Hash, _Pred, _Alloc > >::_M_invalidate_local_if (_Predicate __pred) [inherited]

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal ilocal iterators nested in the safe ones.

4.453.2.15 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.453.2.16 void __gnu_debug::_Safe_unordered_container_base::_M_swap (_Safe_unordered_container_base & __x) [protected], [inherited]

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.453.2.17 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.453.3 Member Data Documentation

4.453.3.1 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.453.3.2 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]

The list of constant local iterators that reference this container.

Definition at line 131 of file safe_unordered_base.h.

4.453.3.3 _Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.453.3.4 _Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]

The list of mutable local iterators that reference this container.

Definition at line 128 of file safe_unordered_base.h.

4.453.3.5 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

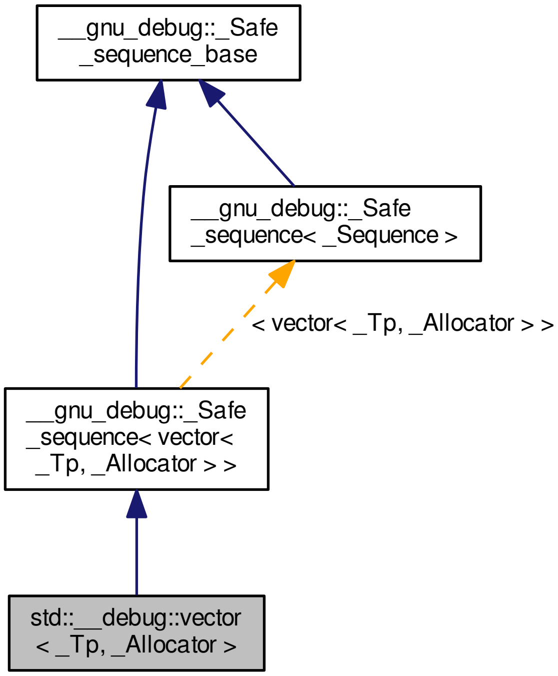
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/unordered_set

4.454 std::__debug::vector< _Tp, _Allocator > Class Template Reference

Inheritance diagram for std::__debug::vector< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, vector > **const_iterator**
- typedef _Base::const_pointer **const_pointer**

- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator
  < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef
  __gnu_debug::_Safe_iterator
  < _Base_iterator, vector > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **vector** (const _Allocator &__a=_Allocator())
- **vector** (size_type __n, const _Allocator &__a=_Allocator())
- **vector** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  vector (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **vector** (const vector &__x)
- vector (const _Base &__x)
- **vector** (vector &&__x) noexcept
- **vector** (const vector &__x, const allocator_type &__a)
- **vector** (vector &&__x, const allocator_type &__a)
- **vector** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- void _M_attach (_Safe_iterator_base ∗__it, bool __constant)
- void _M_attach_single (_Safe_iterator_base ∗__it, bool __constant) throw ()
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void _M_detach (_Safe_iterator_base ∗__it)
- void _M_detach_single (_Safe_iterator_base ∗__it) throw ()
- void _M_invalidate_all () const
- void _M_invalidate_if (_Predicate __pred)
- void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__u)
- void **assign** (initializer_list< value_type > __l)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- size_type **capacity** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept

- template<typename... _Args>
  iterator **emplace** (iterator __position, _Args &&...__args)
- template<typename... _Args>
  void **emplace_back** (_Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, const _Tp &__x)
- template<typename _Up = _Tp>
  __gnu_cxx::__enable_if
  <!std::__are_same< _Up, bool >
  ::__value, iterator >::__type **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __position, initializer_list< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- vector & **operator=** (const vector &__x)
- vector & **operator=** (vector &&__x) noexcept(_Alloc_traits
- vector & **operator=** (initializer_list< value_type > __l)
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- void **pop_back** ()
- void **push_back** (const _Tp &__x)
- template<typename _Up = _Tp>
  __gnu_cxx::__enable_if
  <!std::__are_same< _Up, bool >
  ::__value, void >::__type **push_back** (_Tp &&__x)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **reserve** (size_type __n)
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **shrink_to_fit** ()
- void **swap** (vector &__x) noexcept(_Alloc_traits

Public Attributes

- _Safe_iterator_base ∗ _M_const_iterators
- _Safe_iterator_base ∗ _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all ()
- void _M_detach_singular ()
- __gnu_cxx::__mutex & _M_get_mutex () throw ()
- void _M_revalidate_singular ()
- void _M_swap (_Safe_sequence_base &__x)

4.454.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__debug::vector< _Tp, _Allocator >

Class std::vector with safety/checking/debug instrumentation.

Definition at line 44 of file debug/vector.

4.454.2 Constructor & Destructor Documentation

4.454.2.1 template<typename _Tp, typename _Allocator = std::allocator<_Tp>> std::__debug::vector< _Tp, _Allocator >::vector (const _Base & *__x*) [inline]

Construction from a release-mode vector.

Definition at line 115 of file debug/vector.

4.454.3 Member Function Documentation

4.454.3.1 void __gnu_debug::_Safe_sequence_base::_M_attach (_Safe_iterator_base * *__it,* bool *__constant*) [inherited]

Attach an iterator to this sequence.

4.454.3.2 void __gnu_debug::_Safe_sequence_base::_M_attach_single (_Safe_iterator_base * *__it,* bool *__constant*) throw () [inherited]

Likewise but not thread safe.

4.454.3.3 void __gnu_debug::_Safe_sequence_base::_M_detach (_Safe_iterator_base * *__it*) [inherited]

Detach an iterator from this sequence

4.454.3.4 void __gnu_debug::_Safe_sequence_base::_M_detach_all ( ) [protected], [inherited]

Detach all iterators, leaving them singular.

Referenced by __gnu_debug::_Safe_sequence_base::∼_Safe_sequence_base().

4.454.3.5 void __gnu_debug::_Safe_sequence_base::_M_detach_single (_Safe_iterator_base * *__it*) throw () [inherited]

Likewise but not thread safe.

4.454.3.6 void __gnu_debug::_Safe_sequence_base::_M_detach_singular ( ) [protected], [inherited]

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

4.454.3.7 __gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex ( ) throw () [protected], [inherited]

For use in _Safe_sequence.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.454.3.8 void __gnu_debug::_Safe_sequence_base::_M_invalidate_all ( ) const [inline], [inherited]

Invalidates all iterators.

Definition at line 233 of file safe_base.h.

References __gnu_debug::_Safe_sequence_base::_M_version.

4.454.3.9 void __gnu_debug::_Safe_sequence< vector< _Tp, _Allocator > >::_M_invalidate_if (_Predicate __pred) [inherited]

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.454.3.10 void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular ( ) [protected], [inherited]

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.454.3.11 void __gnu_debug::_Safe_sequence_base::_M_swap (_Safe_sequence_base & __x) [protected], [inherited]

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.454.3.12 void __gnu_debug::_Safe_sequence< vector< _Tp, _Allocator > >::_M_transfer_from_if (_Safe_sequence< vector< _Tp, _Allocator > > & __from, _Predicate __pred) [inherited]

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

4.454.4 Member Data Documentation

4.454.4.1 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]

The list of constant iterators that reference this container.

Definition at line 184 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.454.4.2 _Safe_iterator_base∗ __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]

The list of mutable iterators that reference this container.

Definition at line 181 of file safe_base.h.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

4.454.4.3 unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 187 of file safe_base.h.

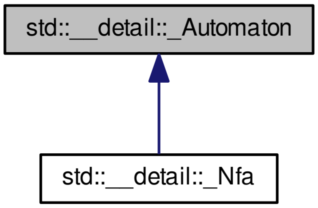
Referenced by __gnu_debug::_Safe_sequence_base::_M_invalidate_all(), and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

The documentation for this class was generated from the following file:

- debug/vector

4.455 std::__detail::_Automaton Class Reference

Inheritance diagram for std::__detail::_Automaton:



Public Types

- typedef unsigned int **_SizeT**

Public Member Functions

- virtual _SizeT **_M_sub_count** () const =0

4.455.1 Detailed Description

Base class for, um, automata. Could be an NFA or a DFA. Your choice.

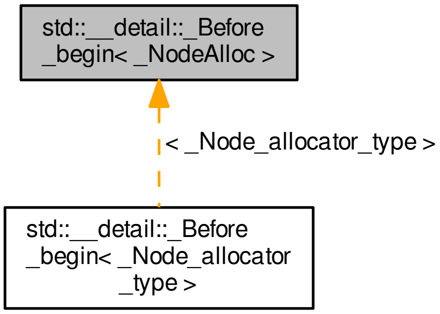
Definition at line 43 of file regex_nfa.h.

The documentation for this class was generated from the following file:

- regex_nfa.h

4.456 std::__detail::_Before_begin< _NodeAlloc > Struct Template Reference

Inheritance diagram for std::__detail::_Before_begin< _NodeAlloc >:



Public Member Functions

- **_Before_begin** (const _Before_begin &)=default
- **_Before_begin** (_Before_begin &&)=default
- template<typename _Alloc >
  _Before_begin (_Alloc &&__a)

Public Attributes

- _Hash_node_base **_M_node**

4.456.1 Detailed Description

template<typename _NodeAlloc>struct std::__detail::_Before_begin< _NodeAlloc >

This type is to combine a _Hash_node_base instance with an allocator instance through inheritance to benefit from EBO when possible.

Definition at line 1652 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.457 std::__detail::_CharMatcher< _InIterT, _TraitsT > Struct Template Reference

Public Types

- typedef _TraitsT::char_type **char_type**

Public Member Functions

- **_CharMatcher** (char_type __c, const _TraitsT &__t=_TraitsT())
- bool **operator()** (const _PatternCursor &__pc) const

Public Attributes

- char_type **_M_c**
- const _TraitsT & **_M_traits**

4.457.1 Detailed Description

template<typename _InIterT, typename _TraitsT>struct std::__detail::_CharMatcher< _InIterT, _TraitsT >

Matches a single character.

Definition at line 129 of file regex_nfa.h.

The documentation for this struct was generated from the following file:

- regex_nfa.h

4.458 std::__detail::_Compiler< _InIter, _TraitsT > Class Template Reference

Public Types

- typedef std::iterator_traits
  < _InIter >::value_type **_CharT**
- typedef
  regex_constants::syntax_option_type **_FlagT**
- typedef _InIter **_IterT**
- typedef std::basic_string< _CharT > **_StringT**

Public Member Functions

- **_Compiler** (const _InIter &__b, const _InIter &__e, _TraitsT &__traits, _FlagT __flags)
- const _Nfa & **_M_nfa** () const

4.458.1 Detailed Description

template<typename _InIter, typename _TraitsT>class std::__detail::_Compiler< _InIter, _TraitsT >

Builds an NFA from an input iterator interval.

Definition at line 634 of file regex_compiler.h.

The documentation for this class was generated from the following file:

- regex_compiler.h

4.459 std::__detail::_Default_ranged_hash Struct Reference

4.459.1 Detailed Description

Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that h(k, N) = h2(h1(k), N), but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.

Definition at line 353 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.460 std::__detail::_EndTagger< _FwdIterT, _TraitsT > Struct Template Reference

Public Member Functions

- **_EndTagger** (int __i)
- void **operator()** (const _PatternCursor &__pc, _Results &__r)

Public Attributes

- int **_M_index**
- _FwdIterT **_M_pos**

4.460.1 Detailed Description

template<typename _FwdIterT, typename _TraitsT>struct std::__detail::_EndTagger< _FwdIterT, _TraitsT >

End state tag.

Definition at line 104 of file regex_nfa.h.

The documentation for this struct was generated from the following file:

- regex_nfa.h

4.461 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false > Struct Template Reference

Static Public Member Functions

- static bool **_S_equals** (const _Equal &__eq, const _ExtractKey &__extract, const _Key &__k, _HashCodeType, _Hash_node< _Value, false > ∗__n)

4.461.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >

Specialization.

Definition at line 1172 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.462 std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true > Struct Template Reference

Static Public Member Functions

- static bool **_S_equals** (const _Equal &__eq, const _ExtractKey &__extract, const _Key &__k, _HashCodeType __c, _Hash_node< _Value, true > ∗__n)

4.462.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _HashCodeType>struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >

Specialization.

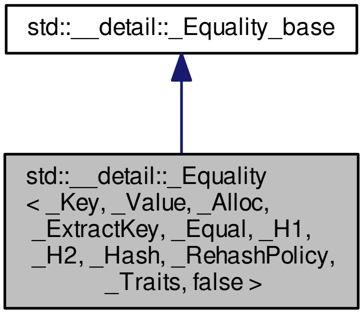
Definition at line 1161 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.463 std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference

Inheritance diagram for std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >:



Public Types

- using **__hashtable** = _Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Public Member Functions

- bool **_M_equal** (const __hashtable &) const

Static Protected Member Functions

- template<typename _Uiterator >
  static bool **_S_is_permutation** (_Uiterator, _Uiterator, _Uiterator)

4.463.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

Specialization.

Definition at line 1604 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.464 std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference

Public Types

- using **__hashtable** = _Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Public Member Functions

- bool **_M_equal** (const __hashtable &) const

4.464.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

Specialization.

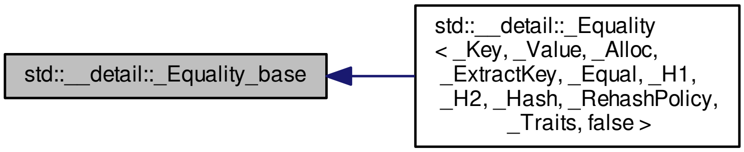
Definition at line 1566 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.465 std::__detail::_Equality_base Struct Reference

Inheritance diagram for std::__detail::_Equality_base:



Static Protected Member Functions

- template<typename _Uiterator >
  static bool **_S_is_permutation** (_Uiterator, _Uiterator, _Uiterator)

4.465.1 Detailed Description

struct _Equality_base.

Common types and functions for class _Equality.

Definition at line 1492 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.466 std::__detail::_Grep_matcher Class Reference

Public Member Functions

- **_Grep_matcher** (_PatternCursor &__p, _Results &__r, const _AutomatonPtr &__automaton, regex_constants::match_flag_type __flags)

4.466.1 Detailed Description

Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.

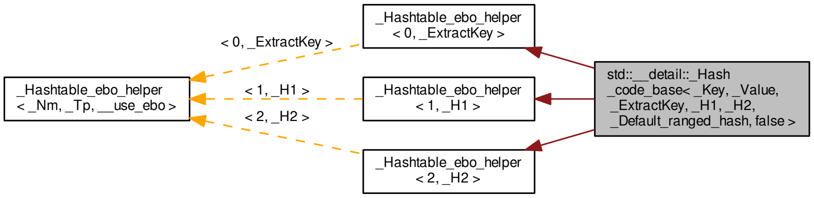
Definition at line 110 of file regex_grep_matcher.h.

The documentation for this class was generated from the following files:

- regex_grep_matcher.h
- regex_grep_matcher.tcc

4.467 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false > Struct Template Reference

Inheritance diagram for std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >:



Public Types

- typedef _H1 **hasher**

Public Member Functions

- hasher **hash_function** () const

Protected Types

- typedef std::size_t **__hash_code**
- typedef _Hash_node< _Value, false > **__node_type**

Protected Member Functions

- **_Hash_code_base** (const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const _Default_ranged_hash &)
- std::size_t **_M_bucket_index** (const _Key &, __hash_code __c, std::size_t __n) const
- std::size_t **_M_bucket_index** (const __node_type ∗__p, std::size_t __n) const
- void **_M_copy_code** (__node_type ∗, const __node_type ∗) const
- const _ExtractKey & **_M_extract** () const
- _ExtractKey & **_M_extract** ()
- const _H1 & **_M_h1** () const
- _H1 & **_M_h1** ()
- const _H2 & **_M_h2** () const
- _H2 & **_M_h2** ()
- __hash_code **_M_hash_code** (const _Key &__k) const
- void **_M_store_code** (__node_type ∗, __hash_code) const
- void **_M_swap** (_Hash_code_base &__x)

4.467.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >

Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.

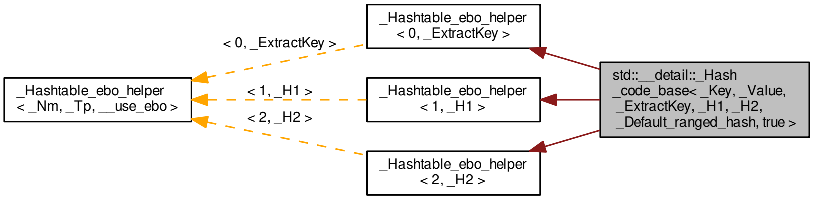
Definition at line 987 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.468 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true > Struct Template Reference

Inheritance diagram for std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >:



Public Types

- typedef _H1 **hasher**

Public Member Functions

- hasher **hash_function** () const

Protected Types

- typedef std::size_t **__hash_code**
- typedef _Hash_node< _Value, true > **__node_type**

Protected Member Functions

- **_Hash_code_base** (const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const _Default_ranged_hash &)
- std::size_t **_M_bucket_index** (const _Key &, __hash_code __c, std::size_t __n) const
- std::size_t **_M_bucket_index** (const __node_type *__p, std::size_t __n) const

- void **_M_copy_code** (__node_type ∗__to, const __node_type ∗__from) const
- const _ExtractKey & **_M_extract** () const
- _ExtractKey & **_M_extract** ()
- const _H1 & **_M_h1** () const
- _H1 & **_M_h1** ()
- const _H2 & **_M_h2** () const
- _H2 & **_M_h2** ()
- __hash_code **_M_hash_code** (const _Key &__k) const
- void **_M_store_code** (__node_type ∗__n, __hash_code __c) const
- void **_M_swap** (_Hash_code_base &__x)

Friends

- struct **_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >**

4.468.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >

Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.

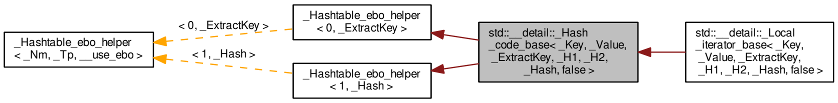
Definition at line 1070 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.469 std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference

Inheritance diagram for std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >:



Protected Types

- typedef void ∗ **__hash_code**
- typedef _Hash_node< _Value, false > **__node_type**

Protected Member Functions

- **_Hash_code_base** (const _ExtractKey &__ex, const _H1 &, const _H2 &, const _Hash &__h)
- std::size_t **_M_bucket_index** (const _Key &__k, __hash_code, std::size_t __n) const
- std::size_t **_M_bucket_index** (const __node_type ∗__p, std::size_t __n) const
- void **_M_copy_code** (__node_type ∗, const __node_type ∗) const
- const _ExtractKey & **_M_extract** () const
- _ExtractKey & **_M_extract** ()
- __hash_code **_M_hash_code** (const _Key &__key) const
- const _Hash & **_M_ranged_hash** () const
- _Hash & **_M_ranged_hash** ()
- void **_M_store_code** (__node_type ∗, __hash_code) const
- void **_M_swap** (_Hash_code_base &__x)

4.469.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

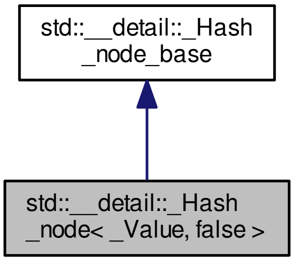
Definition at line 913 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.470 std::__detail::_Hash_node< _Value, false > Struct Template Reference

Inheritance diagram for std::__detail::_Hash_node< _Value, false >:



Public Member Functions

- template<typename... _Args>
  _Hash_node (_Args &&...__args)
- _Hash_node ∗ **_M_next** () const

Public Attributes

- _Hash_node_base ∗ **_M_nxt**
- _Value **_M_v**

4.470.1 Detailed Description

template<typename _Value>struct std::__detail::_Hash_node< _Value, false >

Specialization for nodes without caches, struct _Hash_node.

Base class is __detail::_Hash_node_base.

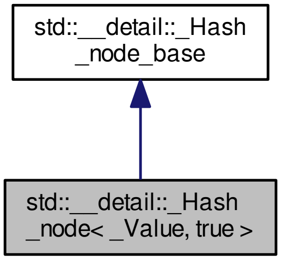
Definition at line 189 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.471 std::__detail::_Hash_node< _Value, true > Struct Template Reference

Inheritance diagram for std::__detail::_Hash_node< _Value, true >:



Public Member Functions

- template<typename... _Args>
  _Hash_node (_Args &&...__args)
- _Hash_node ∗ **_M_next** () const

Public Attributes

- std::size_t **_M_hash_code**
- _Hash_node_base ∗ **_M_nxt**
- _Value **_M_v**

4.471.1 Detailed Description

template<typename _Value>struct std::__detail::_Hash_node< _Value, true >

Specialization for nodes with caches, struct _Hash_node.

Base class is __detail::_Hash_node_base.

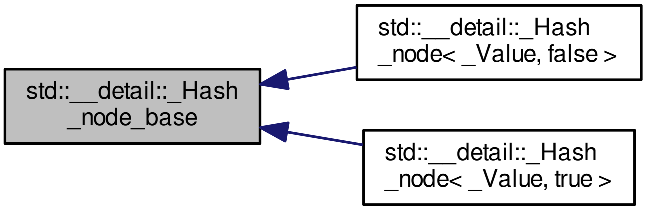
Definition at line 170 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.472 std::__detail::_Hash_node_base Struct Reference

Inheritance diagram for std::__detail::_Hash_node_base:



Public Member Functions

- **_Hash_node_base** (_Hash_node_base ∗__next)

Public Attributes

- _Hash_node_base ∗ **_M_nxt**

4.472.1 Detailed Description

struct _Hash_node_base

Nodes, used to wrap elements stored in the hash table. A policy template parameter of class template _Hashtable controls whether nodes also store a hash code. In some cases (e.g. strings) this may be a performance win.

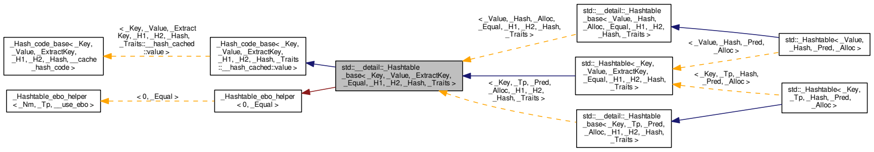
Definition at line 149 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.473 std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits > Struct Template Reference

Inheritance diagram for std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >:



Public Types

- using **__constant_iterators** = typename __traits_type::__constant_iterators
- using **__hash_cached** = typename __traits_type::__hash_cached
- using **__hash_code** = typename __hash_code_base::__hash_code
- using **__hash_code_base** = _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __hash_cached::value >
- using **__iconv_type** = typename std::conditional< __unique_keys::value, _Select1st, _Identity >::type
- using **__ireturn_type** = typename std::conditional< __unique_keys::value, std::pair< iterator, bool >, iterator >::type
- using **__node_type** = typename __hash_code_base::__node_type
- using **__traits_type** = _Traits
- using **__unique_keys** = typename __traits_type::__unique_keys
- using **const_iterator** = __detail::_Node_const_iterator< value_type, __constant_iterators::value, __hash_cached::value >
- using **const_local_iterator** = __detail::_Local_const_iterator< key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __constant_iterators::value, __hash_cached::value >
- typedef std::ptrdiff_t **difference_type**
- using **iterator** = __detail::_Node_iterator< value_type, __constant_iterators::value, __hash_cached::value >
- typedef _Equal **key_equal**
- typedef _Key **key_type**
- using **local_iterator** = __detail::_Local_iterator< key_type, value_type, _ExtractKey, _H1, _H2, _Hash, __constant_iterators::value, __hash_cached::value >
- typedef std::size_t **size_type**
- typedef _Value **value_type**

Protected Types

- using **__bucket_type** = __node_base ∗
- using **__node_base** = __detail::_Hash_node_base

Protected Member Functions

- **_Hashtable_base** (const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const _Hash &__hash, const _Equal &__eq)
- const _Equal & **_M_eq** () const

- _Equal & **_M_eq** ()
- bool **_M_equals** (const _Key &__k, __hash_code __c, __node_type ∗__n) const
- void **_M_swap** (_Hashtable_base &__x)

4.473.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _Traits>struct std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >

Primary class template _Hashtable_base.

Helper class adding management of _Equal functor to _Hash_code_base type.

Base class templates are:

- __detail::_Hash_code_base
- __detail::_Hashtable_ebo_helper

Definition at line 1402 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.474 std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false > Struct Template Reference

Public Member Functions

- **_Hashtable_ebo_helper** (const _Tp &__tp)

Static Public Member Functions

- static const _Tp & **_S_cget** (const _Hashtable_ebo_helper &__eboh)
- static _Tp & **_S_get** (_Hashtable_ebo_helper &__eboh)

4.474.1 Detailed Description

template<int _Nm, typename _Tp>struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >

Specialization not using EBO.

Definition at line 854 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.475 std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true > Struct Template Reference

Inherits _Tp.

Public Member Functions

- **_Hashtable_ebo_helper** (const _Tp &__tp)

Static Public Member Functions

- static const _Tp & **_S_cget** (const _Hashtable_ebo_helper &__eboh)
- static _Tp & **_S_get** (_Hashtable_ebo_helper &__eboh)

4.475.1 Detailed Description

template<int _Nm, typename _Tp>struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >

Specialization using EBO.

Definition at line 835 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.476 std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys > Struct Template Reference

Public Types

- template<bool _Cond>
  using **__bool_constant** = integral_constant< bool, _Cond >
- using **__constant_iterators** = __bool_constant< _Constant_iterators >
- using **__hash_cached** = __bool_constant< _Cache_hash_code >
- using **__unique_keys** = __bool_constant< _Unique_keys >

4.476.1 Detailed Description

template<bool _Cache_hash_code, bool _Constant_iterators, bool _Unique_keys>struct std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >

struct _Hashtable_traits

Important traits for hash tables.

Template Parameters

_Cache_hash_code	Boolean value. True if the value of the hash function is stored along with the value. This is a time-space tradeoff. Storing it may improve lookup speed by reducing the number of times we need to call the _Equal function.
_Constant_iterators	Boolean value. True if iterator and const_iterator are both constant iterator types. This is true for unordered_set and unordered_multiset, false for unordered_map and unordered_multimap.
_Unique_keys	Boolean value. True if the return value of _Hashtable::count(k) is always at most one, false if it may be an arbitrary number. This is true for unordered_set and unordered_map, false for unordered_multiset and unordered_multimap.

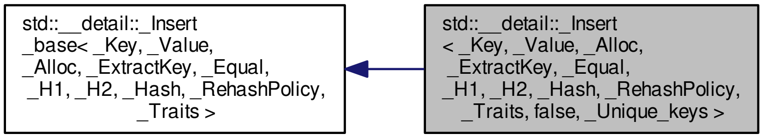
Definition at line 131 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.477 std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys > Struct Template Reference

Inheritance diagram for std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >:



Public Types

- using **__base_type** = _Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- using **__hashtable** = typename __base_type::__hashtable
- using **__hashtable_base** = _Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using **__iconv_type** = typename __base_type::__iconv_type
- using **__ireturn_type** = typename __base_type::__ireturn_type
- template<typename _Pair >
  using **__is_cons** = std::is_constructible< value_type, _Pair && >
- using **__unique_keys** = typename __base_type::__unique_keys
- template<typename _Pair >
  using **_IFcons** = std::enable_if< __is_cons< _Pair >::value >
- template<typename _Pair >
  using **_IFconsp** = typename _IFcons< _Pair >::type
- using **const_iterator** = typename __base_type::const_iterator
- using **iterator** = typename __base_type::iterator
- using **size_type** = typename __hashtable_base::size_type
- using **value_type** = typename __base_type::value_type

Public Member Functions

- __hashtable & **_M_conjure_hashtable** ()
- __ireturn_type **insert** (const value_type &__v)
- iterator **insert** (const_iterator, const value_type &__v)
- void **insert** (initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)

- template<typename _Pair , typename = _IFconsp<_Pair>>
  __ireturn_type **insert** (_Pair &&__v)
- template<typename _Pair , typename = _IFconsp<_Pair>>
  iterator **insert** (const_iterator, _Pair &&__v)

4.477.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys>struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >

Specialization.

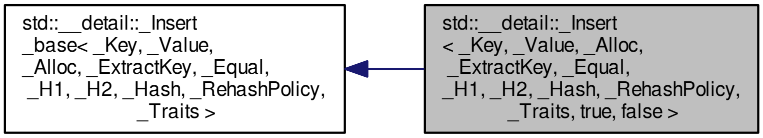
Definition at line 736 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.478 std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false > Struct Template Reference

Inheritance diagram for std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >:



Public Types

- using **__base_type** = _Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- using **__hashtable** = typename __base_type::__hashtable
- using **__hashtable_base** = _Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using **__iconv_type** = typename __hashtable_base::__iconv_type
- using **__ireturn_type** = typename __hashtable_base::__ireturn_type
- using **__unique_keys** = typename __base_type::__unique_keys
- using **const_iterator** = typename __base_type::const_iterator
- using **iterator** = typename __base_type::iterator
- using **size_type** = typename __hashtable_base::size_type
- using **value_type** = typename __base_type::value_type

Public Member Functions

- __hashtable & **_M_conjure_hashtable** ()
- __ireturn_type **insert** (const value_type &__v)
- iterator **insert** (const_iterator, const value_type &__v)
- void **insert** (initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **insert** (value_type &&__v)
- iterator **insert** (const_iterator, value_type &&__v)

4.478.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >

Specialization.

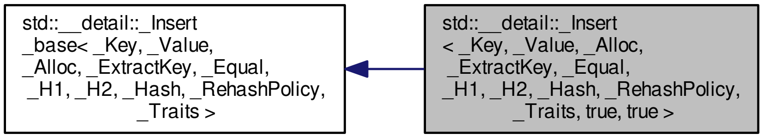
Definition at line 702 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.479 std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true > Struct Template Reference

Inheritance diagram for std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >:



Public Types

- using **__base_type** = _Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- using **__hashtable** = typename __base_type::__hashtable
- using **__hashtable_base** = _Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using **__iconv_type** = typename __hashtable_base::__iconv_type
- using **__ireturn_type** = typename __hashtable_base::__ireturn_type
- using **__unique_keys** = typename __base_type::__unique_keys

- using **const_iterator** = typename __base_type::const_iterator
- using **iterator** = typename __base_type::iterator
- using **size_type** = typename __hashtable_base::size_type
- using **value_type** = typename __base_type::value_type

Public Member Functions

- __hashtable & **_M_conjure_hashtable** ()
- __ireturn_type **insert** (const value_type &__v)
- iterator **insert** (const_iterator, const value_type &__v)
- void **insert** (initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)
- std::pair< iterator, bool > **insert** (value_type &&__v)
- iterator **insert** (const_iterator, value_type &&__v)

4.479.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >

Specialization.

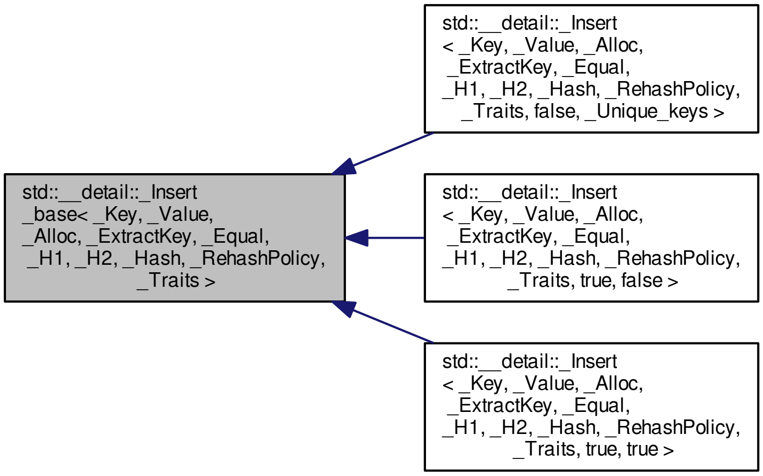
Definition at line 668 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.480 std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Struct Template Reference

Inheritance diagram for std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >:



Public Types

- using **__hashtable** = _Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- using **__hashtable_base** = _Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- using **__iconv_type** = typename __hashtable_base::__iconv_type
- using **__ireturn_type** = typename __hashtable_base::__ireturn_type
- using **__unique_keys** = typename __hashtable_base::__unique_keys
- using **const_iterator** = typename __hashtable_base::const_iterator
- using **iterator** = typename __hashtable_base::iterator
- using **size_type** = typename __hashtable_base::size_type
- using **value_type** = typename __hashtable_base::value_type

Public Member Functions

- __hashtable & **_M_conjure_hashtable** ()
- __ireturn_type **insert** (const value_type &__v)
- iterator **insert** (const_iterator, const value_type &__v)
- void **insert** (initializer_list< value_type > __l)
- template<typename _InputIterator >
  void **insert** (_InputIterator __first, _InputIterator __last)

4.480.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Primary class template _Insert_base.

insert member functions appropriate to all _Hashtables.

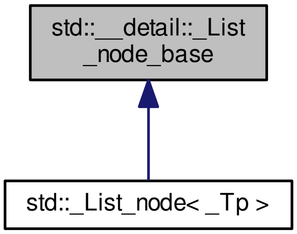
Definition at line 577 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.481 std::__detail::_List_node_base Struct Reference

Inheritance diagram for std::__detail::_List_node_base:



Public Member Functions

- void **_M_hook** (_List_node_base ∗const __position) noexcept
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base ∗const __first, _List_node_base ∗const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base ∗ **_M_next**
- _List_node_base ∗ **_M_prev**

4.481.1 Detailed Description

Common part of a node in the list.

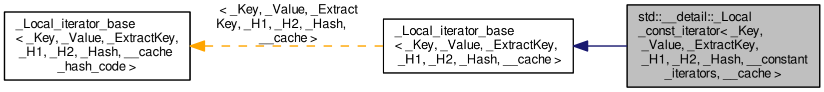
Definition at line 77 of file stl_list.h.

The documentation for this struct was generated from the following file:

- stl_list.h

4.482 std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > Struct Template Reference

Inheritance diagram for std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const _Value ∗ **pointer**
- typedef const _Value & **reference**
- typedef _Value **value_type**

Public Member Functions

- **_Local_const_iterator** (const __hash_code_base &__base, _Hash_node< _Value, __cache > ∗__p, std::size_t __bkt, std::size_t __bkt_count)
- **_Local_const_iterator** (const _Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > &__x)
- reference **operator∗** () const
- _Local_const_iterator & **operator++** ()
- _Local_const_iterator **operator++** (int)
- pointer **operator->** () const

4.482.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>struct std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

local const_iterators

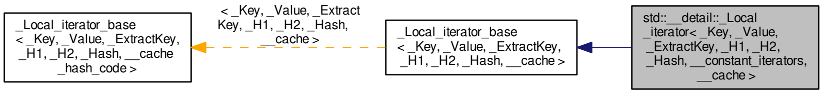
Definition at line 1334 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.483 std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache > Struct Template Reference

Inheritance diagram for std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef std::conditional
  < __constant_iterators, const
  _Value ∗, _Value ∗ >::type **pointer**
- typedef std::conditional
  < __constant_iterators, const
  _Value &, _Value & >::type **reference**
- typedef _Value **value_type**

Public Member Functions

- **_Local_iterator** (const __hash_code_base &__base, _Hash_node< _Value, __cache > ∗__p, std::size_t __bkt, std::size_t __bkt_count)
- reference **operator∗** () const
- _Local_iterator & **operator++** ()
- _Local_iterator **operator++** (int)
- pointer **operator->** () const

4.483.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_iterators, bool __cache>struct std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

local iterators

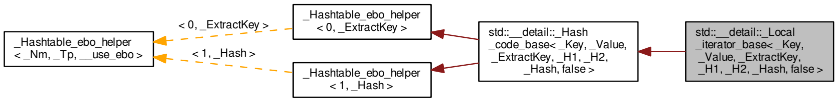
Definition at line 1279 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.484 std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference

Inheritance diagram for std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >:



Public Member Functions

- **_Local_iterator_base** (const __hash_code_base &__base, _Hash_node< _Value, false > ∗__p, std::size_t __bkt, std::size_t __bkt_count)
- void **_M_incr** ()

Public Attributes

- std::size_t **_M_bucket**
- std::size_t **_M_bucket_count**
- _Hash_node< _Value, false > ∗ **_M_cur**

Protected Types

- using **__hash_code_base** = _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

Private Types

- typedef void ∗ **__hash_code**
- typedef _Hash_node< _Value, false > **__node_type**

Private Member Functions

- std::size_t **_M_bucket_index** (const _Key &__k, __hash_code, std::size_t __n) const
- std::size_t **_M_bucket_index** (const __node_type ∗__p, std::size_t __n) const
- void **_M_copy_code** (__node_type ∗, const __node_type ∗) const
- const _ExtractKey & **_M_extract** () const
- _ExtractKey & **_M_extract** ()
- __hash_code **_M_hash_code** (const _Key &__key) const
- const _Hash & **_M_ranged_hash** () const
- _Hash & **_M_ranged_hash** ()
- void **_M_store_code** (__node_type ∗, __hash_code) const
- void **_M_swap** (_Hash_code_base &__x)

4.484.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

Specialization.

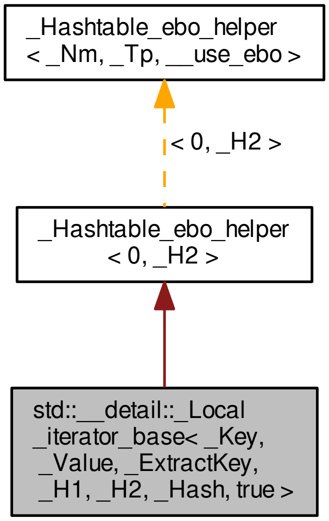
Definition at line 1223 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.485 std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true > Struct Template Reference

Inheritance diagram for std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >:



Public Member Functions

- **_Local_iterator_base** (const __hash_code_base &__base, _Hash_node< _Value, true > ∗__p, std::size_t __bkt, std::size_t __bkt_count)
- void **_M_incr** ()

Public Attributes

- std::size_t **_M_bucket**
- std::size_t **_M_bucket_count**

- _Hash_node< _Value, true > ∗ **_M_cur**

Protected Types

- using **__base_type** = _Hashtable_ebo_helper< 0, _H2 >
- using **__hash_code_base** = _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >

4.485.1 Detailed Description

template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >

Specialization.

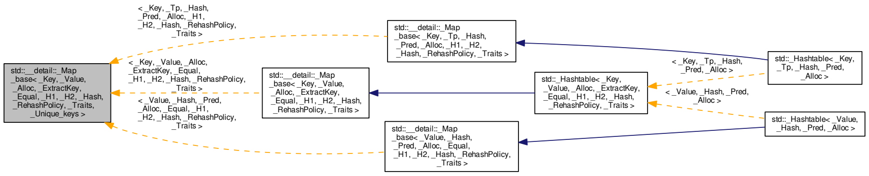
Definition at line 1184 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.486 std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference

Inheritance diagram for std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >:



4.486.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>struct std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >

Primary class template _Map_base.

If the hashtable has a value type of the form pair<T1, T2> and a key extraction policy (_ExtractKey) that returns the first part of the pair, the hashtable gets a mapped_type typedef. If it satisfies those criteria and also has unique keys, then it also gets an operator[].

Definition at line 424 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.487 std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference

Public Types

- using **mapped_type** = typename std::tuple_element< 1, _Pair >::type

4.487.1 Detailed Description

template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

Partial specialization, __unique_keys set to false.

Definition at line 430 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.488 std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference

Public Types

- using **iterator** = typename __hashtable_base::iterator
- using **key_type** = typename __hashtable_base::key_type
- using **mapped_type** = typename std::tuple_element< 1, _Pair >::type

Public Member Functions

- mapped_type & **at** (const key_type &__k)
- const mapped_type & **at** (const key_type &__k) const
- mapped_type & **operator[]** (const key_type &__k)
- mapped_type & **operator[]** (key_type &&__k)

4.488.1 Detailed Description

template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

Partial specialization, __unique_keys set to true.

Definition at line 440 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.489 std::__detail::_Mod_range_hashing Struct Reference

Public Types

- typedef std::size_t **first_argument_type**
- typedef std::size_t **result_type**
- typedef std::size_t **second_argument_type**

Public Member Functions

- result_type **operator()** (first_argument_type __num, second_argument_type __den) const

4.489.1 Detailed Description

Default range hashing function: use division to fold a large number into the range [0, N).

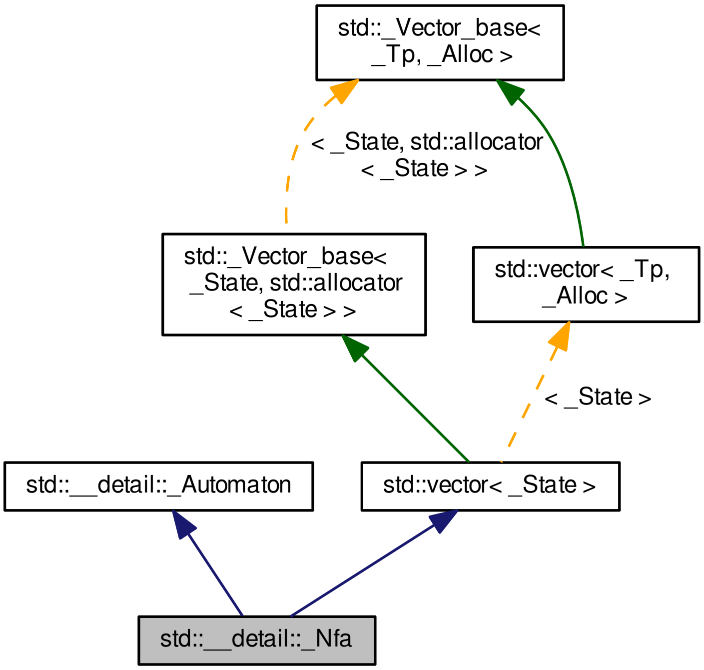
Definition at line 337 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.490 std::__detail::_Nfa Class Reference

Inheritance diagram for std::__detail::_Nfa:



Public Types

- typedef regex_constants::syntax_option_type **_FlagT**
- typedef unsigned int **_SizeT**
- typedef _State **_StateT**
- typedef std::allocator< _State > **allocator_type**
- typedef __gnu_cxx::__normal_iterator < const_pointer, vector > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator < pointer, vector > **iterator**

- typedef _Base::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _State **value_type**

Public Member Functions

- **_Nfa** (_FlagT __f)
- const _StateSet & **_M_final_states** () const
- _StateIdT **_M_insert_accept** ()
- _StateIdT **_M_insert_alt** (_StateIdT __next, _StateIdT __alt)
- _StateIdT **_M_insert_matcher** (_Matcher __m)
- _StateIdT **_M_insert_subexpr_begin** (const _Tagger &__t)
- _StateIdT **_M_insert_subexpr_end** (unsigned int __i, const _Tagger &__t)
- _FlagT **_M_options** () const
- _StateIdT **_M_start** () const
- _SizeT **_M_sub_count** () const
- void assign (size_type __n, const value_type &__val)
- void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- reference at (size_type __n)
- const_reference at (size_type __n) const
- reference back ()
- const_reference back () const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- size_type capacity () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- _State ∗ data () noexcept
- const _State ∗ **data** () const noexcept
- iterator emplace (iterator __position, _Args &&...__args)
- void **emplace_back** (_Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- reference front ()
- const_reference front () const
- iterator insert (iterator __position, const value_type &__x)
- iterator insert (iterator __position, value_type &&__x)
- void insert (iterator __position, initializer_list< value_type > __l)
- void insert (iterator __position, size_type __n, const value_type &__x)
- void insert (iterator __position, _InputIterator __first, _InputIterator __last)

- size_type max_size () const noexcept
- reference operator[] (size_type __n)
- const_reference operator[] (size_type __n) const
- void pop_back ()
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- void reserve (size_type __n)
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void shrink_to_fit ()
- size_type size () const noexcept
- void swap (vector &__x) noexcept(_Alloc_traits

Protected Member Functions

- pointer **_M_allocate** (size_t __n)
- pointer _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- size_type **_M_check_len** (size_type __n, const char $*$__s) const
- void **_M_deallocate** (pointer __p, size_t __n)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_emplace_back_aux** (_Args &&...__args)
- void **_M_erase_at_end** (pointer __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- void **_M_initialize_dispatch** (_Integer __n, _Integer __value, __true_type)
- void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_insert_aux** (iterator __position, _Args &&...__args)
- void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void _M_range_check (size_type __n) const
- void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- bool **_M_shrink_to_fit** ()
- allocator_type **get_allocator** () const noexcept

Protected Attributes

- _Vector_impl **_M_impl**

4.490.1 Detailed Description

struct _Nfa

A collection of all states making up an NFA.

An NFA is a 4-tuple M = (K, S, s, F), where K is a finite set of states, S is the alphabet of the NFA, s is the initial state, F is a set of final (accepting) states.

This NFA class is templated on S, a type that will hold values of the underlying alphabet (without regard to semantics of that alphabet). The other elements of the tuple are generated during construction of the NFA and are available through accessor member functions.

Definition at line 269 of file regex_nfa.h.

4.490.2 Member Function Documentation

4.490.2.1 pointer std::vector< _State , std::allocator< _State > >::_M_allocate_and_copy (size_type *__n,* _ForwardIterator *__first,* _ForwardIterator *__last*) [inline], [protected], [inherited]

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

Definition at line 1135 of file stl_vector.h.

4.490.2.2 void std::vector< _State , std::allocator< _State > >::_M_range_check (size_type *__n*) const [inline], [protected], [inherited]

Safety check used only from at().

Definition at line 791 of file stl_vector.h.

4.490.2.3 void std::vector< _State , std::allocator< _State > >::assign (size_type *__n,* const value_type & *__val*) [inline], [inherited]

Assigns a given value to a vector.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a vector with *__n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 479 of file stl_vector.h.

4.490.2.4 void std::vector< _State , std::allocator< _State > >::assign (_InputIterator *__first,* _InputIterator *__last*) [inline], [inherited]

Assigns a range to a vector.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a vector with copies of the elements in the range [__first,__last).

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 498 of file stl_vector.h.

4.490.2.5 void std::vector< _State , std::allocator< _State > >::assign (initializer_list< value_type > __l) [inline], [inherited]

Assigns an initializer list to a vector.

Parameters

__l	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 524 of file stl_vector.h.

4.490.2.6 reference std::vector< _State , std::allocator< _State > >::at (size_type __n) [inline], [inherited]

Provides access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

Definition at line 810 of file stl_vector.h.

4.490.2.7 const_reference std::vector< _State , std::allocator< _State > >::at (size_type __n) const [inline], [inherited]

Provides access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

Definition at line 828 of file stl_vector.h.

4.490.2.8 reference **std::vector**< **_State** , **std::allocator**<**_State** > >::back ( ) `[inline]`, `[inherited]`

Returns a read/write reference to the data at the last element of the vector.

Definition at line 855 of file stl_vector.h.

4.490.2.9 const_reference **std::vector**< **_State** , **std::allocator**<**_State** > >::back ( ) const `[inline]`, `[inherited]`

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 863 of file stl_vector.h.

4.490.2.10 iterator **std::vector**< **_State** , **std::allocator**<**_State** > >::begin ( ) `[inline]`, `[noexcept]`, `[inherited]`

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 538 of file stl_vector.h.

4.490.2.11 const_iterator **std::vector**< **_State** , **std::allocator**<**_State** > >::begin ( ) const `[inline]`, `[noexcept]`, `[inherited]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file stl_vector.h.

4.490.2.12 size_type **std::vector**< **_State** , **std::allocator**<**_State** > >::capacity ( ) const `[inline]`, `[noexcept]`, `[inherited]`

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 725 of file stl_vector.h.

4.490.2.13 const_iterator **std::vector**< **_State** , **std::allocator**<**_State** > >::cbegin ( ) const `[inline]`, `[noexcept]`, `[inherited]`

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 611 of file stl_vector.h.

4.490.2.14 const_iterator std::vector< _State , std::allocator< _State > >::cend ( ) const [inline], [noexcept], [inherited]

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file stl_vector.h.

4.490.2.15 void std::vector< _State , std::allocator< _State > >::clear ( ) [inline], [noexcept], [inherited]

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1125 of file stl_vector.h.

4.490.2.16 const_reverse_iterator std::vector< _State , std::allocator< _State > >::crbegin ( ) const [inline], [noexcept], [inherited]

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 629 of file stl_vector.h.

4.490.2.17 const_reverse_iterator std::vector< _State , std::allocator< _State > >::crend ( ) const [inline], [noexcept], [inherited]

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file stl_vector.h.

4.490.2.18 _State ∗ std::vector< _State , std::allocator< _State > >::data ( ) [inline], [noexcept], [inherited]

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

Definition at line 878 of file stl_vector.h.

4.490.2.19 iterator std::vector< _State , std::allocator< _State > >::emplace (iterator *__position*, _Args &&... *__args*) [inherited]

Inserts an object in vector before specified iterator.

Parameters

__position	An iterator into the vector.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

4.490.2.20 bool std::vector< _State , std::allocator< _State > >::empty ( ) const [inline], [noexcept], [inherited]

Returns true if the vector is empty. (Thus begin() would equal end().)

Definition at line 734 of file stl_vector.h.

4.490.2.21 iterator std::vector< _State , std::allocator< _State > >::end ( ) [inline], [noexcept], [inherited]

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file stl_vector.h.

4.490.2.22 const_iterator std::vector< _State , std::allocator< _State > >::end ( ) const [inline], [noexcept], [inherited]

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file stl_vector.h.

4.490.2.23 iterator std::vector< _State , std::allocator< _State > >::erase (iterator *__position*) [inherited]

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.490.2.24 iterator std::vector< _State , std::allocator< _State > >::erase (iterator *__first,* iterator *__last*) [inherited]

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *__last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

4.490.2.25 reference std::vector< _State , std::allocator< _State > >::front ( ) [inline], [inherited]

Returns a read/write reference to the data at the first element of the vector.

Definition at line 839 of file stl_vector.h.

4.490.2.26 const_reference std::vector< _State , std::allocator< _State > >::front ( ) const [inline], [inherited]

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 847 of file stl_vector.h.

4.490.2.27 iterator std::vector< _State , std::allocator< _State > >::insert (iterator *__position*, const value_type & *__x*) [inherited]

Inserts given value into vector before specified iterator.

Parameters

__position	An iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

4.490.2.28 iterator std::vector< _State , std::allocator< _State > >::insert (iterator *__position*, value_type && *__x*) [inline], [inherited]

Inserts given rvalue into vector before specified iterator.

Parameters

__position	An iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 988 of file stl_vector.h.

4.490.2.29 void **std::vector**< **_State , std::allocator**<**_State** > >::insert (iterator *__position,* **initializer_list**< value_type > *__l*) [inline], [inherited]

Inserts an initializer_list into the vector.

Parameters

__position	An iterator into the vector.
__l	An initializer_list.

This function will insert copies of the data in the initializer_list *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1005 of file stl_vector.h.

4.490.2.30 void **std::vector**< **_State , std::allocator**<**_State** > >::insert (iterator *__position,* size_type *__n,* const value_type & *__x*) [inline], [inherited]

Inserts a number of copies of given data into the vector.

Parameters

__position	An iterator into the vector.
__n	Number of elements to be inserted.
__x	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1023 of file stl_vector.h.

4.490.2.31 void **std::vector**< **_State , std::allocator**<**_State** > >::insert (iterator *__position,* _InputIterator *__first,* _InputIterator *__last*) [inline], [inherited]

Inserts a range into the vector.

Parameters

__position	An iterator into the vector.
__first	An input iterator.
__last	An input iterator.

This function will insert copies of the data in the range [__first,__last) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1044 of file stl_vector.h.

4.490.2.32 size_type **std::vector**< **_State , std::allocator**<**_State** > >::max_size ( ) const [inline], [noexcept], [inherited]

Returns the size() of the largest possible vector.

Definition at line 650 of file stl_vector.h.

4.490.2.33 reference **std::vector**< **_State** , **std::allocator**< **_State** > >::operator[] (size_type *__n*) [inline], [inherited]

Subscript access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 770 of file stl_vector.h.

4.490.2.34 const_reference **std::vector**< **_State** , **std::allocator**< **_State** > >::operator[] (size_type *__n*) const [inline], [inherited]

Subscript access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 785 of file stl_vector.h.

4.490.2.35 void **std::vector**< **_State** , **std::allocator**< **_State** > >::pop_back ( ) [inline], [inherited]

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

Definition at line 937 of file stl_vector.h.

4.490.2.36 void **std::vector**< **_State** , **std::allocator**< **_State** > >::push_back (const value_type & *__x*) [inline], [inherited]

Add data to the end of the vector.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 901 of file stl_vector.h.

4.490.2.37 reverse_iterator std::vector< _State , std::allocator<_State > >::rbegin ( ) [inline], [noexcept], [inherited]

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 574 of file stl_vector.h.

4.490.2.38 const_reverse_iterator std::vector< _State , std::allocator<_State > >::rbegin ( ) const [inline], [noexcept], [inherited]

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file stl_vector.h.

4.490.2.39 reverse_iterator std::vector< _State , std::allocator<_State > >::rend ( ) [inline], [noexcept], [inherited]

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file stl_vector.h.

4.490.2.40 const_reverse_iterator std::vector< _State , std::allocator<_State > >::rend ( ) const [inline], [noexcept], [inherited]

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file stl_vector.h.

4.490.2.41 void std::vector< _State , std::allocator<_State > >::reserve (size_type *__n*) [inherited]

Attempt to preallocate enough memory for specified number of elements.

Parameters

__n	Number of elements required.

Exceptions

std::length_error	If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than max_size(), length_error is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

4.490.2.42 void std::vector< _State , std::allocator<_State > >::resize (size_type *__new_size*) [inline], [inherited]

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 664 of file stl_vector.h.

4.490.2.43 void **std::vector**< **_State , std::allocator**< **_State** > >::resize (size_type *__new_size,* const value_type & *__x*) [inline], [inherited]

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.
__x	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 684 of file stl_vector.h.

4.490.2.44 void **std::vector**< **_State , std::allocator**< **_State** > >::shrink_to_fit ( ) [inline], [inherited]

A non-binding request to reduce capacity() to size().

Definition at line 716 of file stl_vector.h.

4.490.2.45 size_type **std::vector**< **_State , std::allocator**< **_State** > >::size ( ) const [inline], [noexcept], [inherited]

Returns the number of elements in the vector.

Definition at line 645 of file stl_vector.h.

4.490.2.46 void **std::vector**< **_State , std::allocator**< **_State** > >::swap (**vector**< _State > & *__x*) [inline], [noexcept], [inherited]

Swaps data with another vector.

Parameters

__x	A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(v1,v2) will feed to this function.

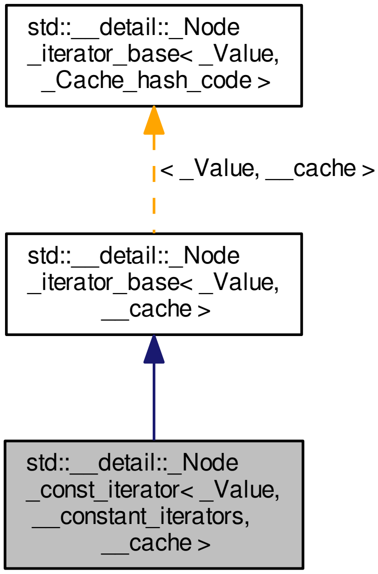
Definition at line 1108 of file stl_vector.h.

The documentation for this class was generated from the following file:

- regex_nfa.h

4.491 std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache > Struct Template Reference

Inheritance diagram for std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const _Value ∗ **pointer**
- typedef const _Value & **reference**
- typedef _Value **value_type**

Public Member Functions

- **_Node_const_iterator** (__node_type ∗__p)
- **_Node_const_iterator** (const _Node_iterator< _Value, __constant_iterators, __cache > &__x)
- void **_M_incr** ()
- reference **operator∗** () const
- _Node_const_iterator & **operator++** ()
- _Node_const_iterator **operator++** (int)
- pointer **operator->** () const

Public Attributes

- __node_type ∗ **_M_cur**

4.491.1 Detailed Description

template<typename _Value, bool __constant_iterators, bool __cache>struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >

Node const_iterators, used to iterate through all the hashtable.

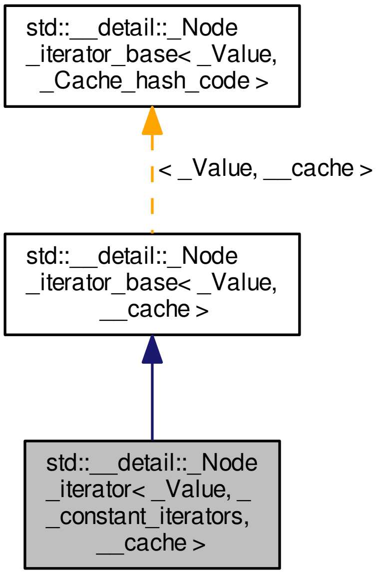
Definition at line 282 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.492 std::__detail::_Node_iterator< _Value, __constant_iterators, __cache > Struct Template Reference

Inheritance diagram for std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- using **pointer** = typename std::conditional< __constant_iterators, const _Value ∗, _Value ∗ >::type
- using **reference** = typename std::conditional< __constant_iterators, const _Value &, _Value & >::type
- typedef _Value **value_type**

Public Member Functions

- **_Node_iterator** (__node_type ∗__p)
- void **_M_incr** ()
- reference **operator∗** () const
- _Node_iterator & **operator++** ()
- _Node_iterator **operator++** (int)
- pointer **operator->** () const

Public Attributes

- __node_type ∗ **_M_cur**

4.492.1 Detailed Description

template<typename _Value, bool __constant_iterators, bool __cache>struct std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >

Node iterators, used to iterate through all the hashtable.

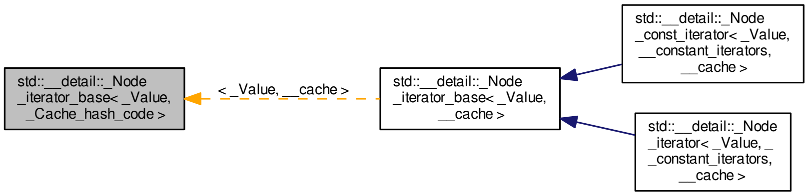
Definition at line 231 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.493 std::__detail::_Node_iterator_base< _Value, _Cache_hash_code > Struct Template Reference

Inheritance diagram for std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >:



Public Types

- using **__node_type** = _Hash_node< _Value, _Cache_hash_code >

Public Member Functions

- **_Node_iterator_base** (__node_type ∗__p)
- void **_M_incr** ()

Public Attributes

- __node_type ∗ **_M_cur**

4.493.1 Detailed Description

template<typename _Value, bool _Cache_hash_code>struct std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >

Base class for node iterators.

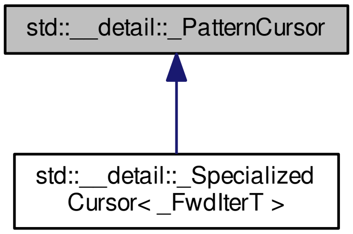
Definition at line 203 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.494 std::__detail::_PatternCursor Struct Reference

Inheritance diagram for std::__detail::_PatternCursor:



Public Member Functions

- virtual bool **_M_at_end** () const =0
- virtual void **_M_next** ()=0

4.494.1 Detailed Description

ABC for pattern matching.

Definition at line 44 of file regex_cursor.h.

The documentation for this struct was generated from the following file:

- regex_cursor.h

4.495 std::__detail::_Prime_rehash_policy Struct Reference

Public Types

- enum { **_S_n_primes** }
- typedef std::size_t **_State**

Public Member Functions

- **_Prime_rehash_policy** (float __z=1.0)
- std::size_t **_M_bkt_for_elements** (std::size_t __n) const
- std::pair< bool, std::size_t > **_M_need_rehash** (std::size_t __n_bkt, std::size_t __n_elt, std::size_t __n_ins) const
- std::size_t **_M_next_bkt** (std::size_t __n) const
- void **_M_reset** (_State __state)
- _State **_M_state** () const
- float **max_load_factor** () const noexcept

Public Attributes

- float **_M_max_load_factor**
- std::size_t **_M_next_resize**

Static Public Attributes

- static const std::size_t **_S_growth_factor**

4.495.1 Detailed Description

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

Definition at line 357 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.496 std::__detail::_RangeMatcher< _InIterT, _TraitsT > Struct Template Reference

Public Types

- typedef _TraitsT::char_type **_CharT**
- typedef std::basic_string< _CharT > **_StringT**

Public Member Functions

- **_RangeMatcher** (bool __is_non_matching, const _TraitsT &__t=_TraitsT())
- void **_M_add_char** (_CharT __c)
- void **_M_add_character_class** (const _StringT &__s)
- void **_M_add_collating_element** (const _StringT &__s)

- void **_M_add_equivalence_class** (const _StringT &__s)
- void **_M_make_range** ()
- bool **operator()** (const _PatternCursor &__pc) const

Public Attributes

- bool **_M_is_non_matching**
- const _TraitsT & **_M_traits**

4.496.1 Detailed Description

template<typename _InIterT, typename _TraitsT>struct std::__detail::_RangeMatcher< _InIterT, _TraitsT >

Matches a character range (bracket expression)

Definition at line 152 of file regex_nfa.h.

The documentation for this struct was generated from the following file:

- regex_nfa.h

4.497 std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits > Struct Template Reference

Public Types

- using **__hashtable** = _Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >

Public Member Functions

- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __z)
- void **reserve** (std::size_t __n)

4.497.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _Traits>struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >

Specialization.

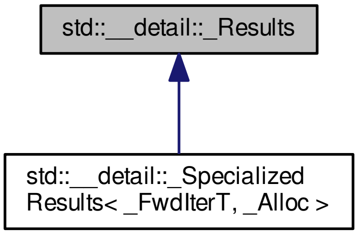
Definition at line 794 of file hashtable_policy.h.

The documentation for this struct was generated from the following file:

- hashtable_policy.h

4.498 std::__detail::_Results Struct Reference

Inheritance diagram for std::__detail::_Results:



Public Member Functions

- virtual void **_M_set_matched** (int __i, bool __is_matched)=0
- virtual void **_M_set_pos** (int __i, int __j, const _PatternCursor &__p)=0

4.498.1 Detailed Description

Provides a generic facade for a templated match_results.

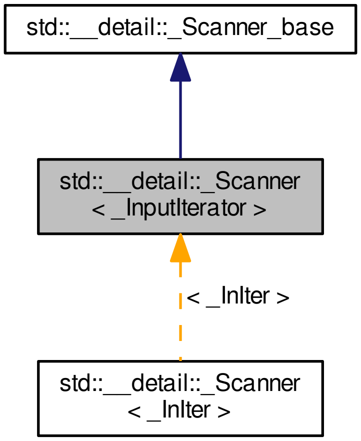
Definition at line 77 of file regex_nfa.h.

The documentation for this struct was generated from the following file:

- regex_nfa.h

4.499 std::__detail::_Scanner< _InputIterator > Class Template Reference

Inheritance diagram for std::__detail::_Scanner< _InputIterator >:



Public Types

- typedef std::iterator_traits
< _IteratorT >::value_type **_CharT**
- typedef const std::ctype< _CharT > **_CtypeT**
- typedef
regex_constants::syntax_option_type **_FlagT**
- typedef _InputIterator **_IteratorT**
- typedef unsigned int **_StateT**
- typedef std::basic_string< _CharT > **_StringT**
- enum _TokenT {
_S_token_anychar, **_S_token_backref**, **_S_token_bracket_begin**, **_S_token_bracket_end**, **_S_token_inverse_class**, **_S_token_char_class_name**, **_S_token_closure0**, **_S_token_closure1**, **_S_token_collelem_multi**, **_S_token_collelem_single**, **_S_token_collsymbol**, **_S_token_comma**, **_S_token_dash**, **_S_token_dup_count**, **_S_token_eof**, **_S_token_equiv_class_name**, **_S_token_interval_begin**, **_S_token_interval_end**, **_S_token_line_begin**, **_S_token_line_end**, **_S_token_opt**, **_S_token_or**, **_S_token_ord_char**, **_S_token_quoted_char**, **_S_token_subexpr_begin**, **_S_token_subexpr_end**, **_S_token_word_begin**, **_S_token_word_end**, **_S_token_unknown** }

Public Member Functions

- **_Scanner** (_IteratorT __begin, _IteratorT __end, _FlagT __flags, std::locale __loc)
- void **_M_advance** ()
- _TokenT **_M_token** () const
- const _StringT & **_M_value** () const

Static Public Attributes

- static constexpr _StateT **_S_state_at_start**
- static constexpr _StateT **_S_state_in_brace**
- static constexpr _StateT **_S_state_in_bracket**

4.499.1 Detailed Description

template<typename _InputIterator>class std::__detail::_Scanner< _InputIterator >

struct _Scanner. Scans an input range for regex tokens.

The _Scanner class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

Definition at line 65 of file regex_compiler.h.

4.499.2 Member Enumeration Documentation

4.499.2.1 template<typename _InputIterator> enum std::__detail::_Scanner::_TokenT

Token types returned from the scanner.

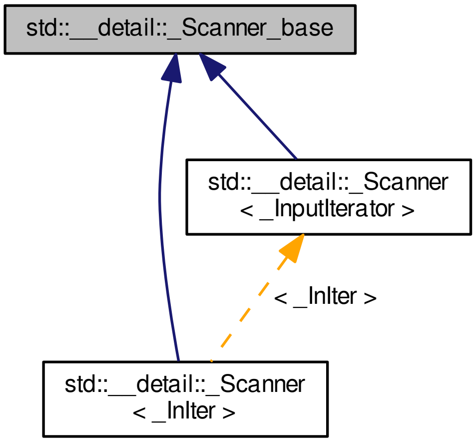
Definition at line 75 of file regex_compiler.h.

The documentation for this class was generated from the following file:

- regex_compiler.h

4.500 std::__detail::_Scanner_base Struct Reference

Inheritance diagram for std::__detail::_Scanner_base:



Public Types

- typedef unsigned int **_StateT**

Static Public Attributes

- static constexpr _StateT **_S_state_at_start**
- static constexpr _StateT **_S_state_in_brace**
- static constexpr _StateT **_S_state_in_bracket**

4.500.1 Detailed Description

Base class for scanner.

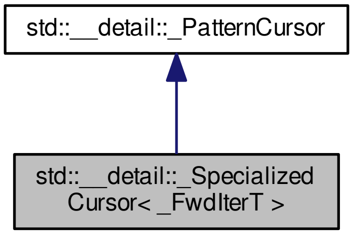
Definition at line 43 of file regex_compiler.h.

The documentation for this struct was generated from the following file:

- regex_compiler.h

4.501 std::__detail::_SpecializedCursor< _FwdIterT > Class Template Reference

Inheritance diagram for std::__detail::_SpecializedCursor< _FwdIterT >:



Public Member Functions

- **_SpecializedCursor** (const _FwdIterT &__b, const _FwdIterT __e)
- bool **_M_at_end** () const
- const _FwdIterT & **_M_begin** () const
- std::iterator_traits < _FwdIterT >::value_type **_M_current** () const
- const _FwdIterT & **_M_end** () const
- void **_M_next** ()
- _FwdIterT **_M_pos** () const

4.501.1 Detailed Description

template<typename _FwdIterT>class std::__detail::_SpecializedCursor< _FwdIterT >

Provides a cursor into the specific target string.

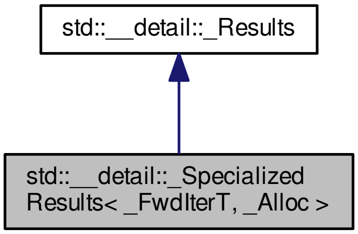
Definition at line 53 of file regex_cursor.h.

The documentation for this class was generated from the following file:

- regex_cursor.h

4.502 std::__detail::_SpecializedResults< _FwdIterT, _Alloc > Class Template Reference

Inheritance diagram for std::__detail::_SpecializedResults< _FwdIterT, _Alloc >:



Public Member Functions

- **_SpecializedResults** (const _Automaton::_SizeT __size, const _SpecializedCursor< _FwdIterT > &__cursor, match_results< _FwdIterT, _Alloc > &__m)
- void **_M_set_matched** (int __i, bool __is_matched)
- void **_M_set_pos** (int __i, int __j, const _PatternCursor &__pc)

4.502.1 Detailed Description

template<typename _FwdIterT, typename _Alloc>class std::__detail::_SpecializedResults< _FwdIterT, _Alloc >

A _Results facade specialized for wrapping a templated match_results.

Definition at line 55 of file regex_grep_matcher.h.

The documentation for this class was generated from the following file:

- regex_grep_matcher.h

4.503 std::__detail::_StartTagger< _FwdIterT, _TraitsT > Struct Template Reference

Public Member Functions

- **_StartTagger** (int __i)
- void **operator()** (const _PatternCursor &__pc, _Results &__r)

Public Attributes

- int **_M_index**

4.503.1 Detailed Description

template<typename _FwdIterT, typename _TraitsT>struct std::__detail::_StartTagger< _FwdIterT, _TraitsT >

Start state tag.

Definition at line 88 of file regex_nfa.h.

The documentation for this struct was generated from the following file:

- regex_nfa.h

4.504 std::__detail::_State Struct Reference

Public Types

- typedef int **_OpcodeT**

Public Member Functions

- **_State** (_OpcodeT __opcode)
- **_State** (const _Matcher &__m)
- **_State** (_OpcodeT __opcode, unsigned int __s, const _Tagger &__t)
- **_State** (_StateIdT __next, _StateIdT __alt)

Public Attributes

- _StateIdT **_M_alt**
- _Matcher **_M_matches**
- _StateIdT **_M_next**
- _OpcodeT **_M_opcode**
- unsigned int **_M_subexpr**
- _Tagger **_M_tagger**

4.504.1 Detailed Description

struct _State

An individual state in an NFA

In this case a "state" is an entry in the NFA definition coupled with its outgoing transition(s). All states have a single outgoing transition, except for accepting states (which have no outgoing transitions) and alt states, which have two outgoing transitions.

Definition at line 211 of file regex_nfa.h.

The documentation for this struct was generated from the following file:

- regex_nfa.h

4.505 std::__detail::_StateSeq Class Reference

Public Member Functions

- **_StateSeq** (_Nfa &__ss, _StateIdT __s, _StateIdT __e=_S_invalid_state_id)
- **_StateSeq** (const _StateSeq &__e1, const _StateSeq &__e2)
- **_StateSeq** (const _StateSeq &__e, _StateIdT __id)
- **_StateSeq** (const _StateSeq &__rhs)
- void **_M_append** (_StateIdT __id)
- void **_M_append** (_StateSeq &__rhs)
- _StateIdT **_M_clone** ()
- _StateIdT **_M_front** () const
- void **_M_push_back** (_StateIdT __id)
- _StateSeq & **operator=** (const _StateSeq &__rhs)

4.505.1 Detailed Description

Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Definition at line 352 of file regex_nfa.h.

The documentation for this class was generated from the following files:

- regex_nfa.h
- regex_nfa.tcc

4.506 std::__exception_ptr::exception_ptr Class Reference

Public Member Functions

- **exception_ptr** (const exception_ptr &) noexcept
- **exception_ptr** (nullptr_t) noexcept
- **exception_ptr** (exception_ptr &&__o) noexcept
- class type_info ∗ **__cxa_exception_type** () const noexcept __attribute__((__pure__))
- **operator bool** () const
- exception_ptr & **operator=** (const exception_ptr &) noexcept
- exception_ptr & **operator=** (exception_ptr &&__o) noexcept
- void **swap** (exception_ptr &) noexcept

Friends

- bool **operator==** (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))
- exception_ptr **std::current_exception** () noexcept
- void **std::rethrow_exception** (exception_ptr)

4.506.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 73 of file exception_ptr.h.

The documentation for this class was generated from the following file:

- exception_ptr.h

4.507 std::__has_iterator_category_helper< _Tp > Class Template Reference

Inherits std::__sfinae_types.

Static Public Attributes

- static constexpr bool **value**

Private Types

- typedef char **__one**

4.507.1 Detailed Description

template<typename _Tp>class std::__has_iterator_category_helper< _Tp >

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the Iterator argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

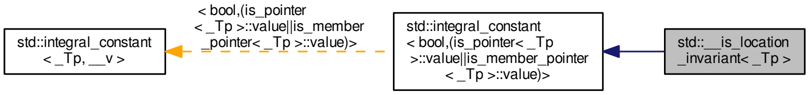
Definition at line 142 of file stl_iterator_base_types.h.

The documentation for this class was generated from the following file:

- stl_iterator_base_types.h

4.508 std::__is_location_invariant< _Tp > Struct Template Reference

Inheritance diagram for std::__is_location_invariant< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.508.1 Detailed Description

template<typename _Tp>struct std::__is_location_invariant< _Tp >

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Also implies a trivial copy constructor and assignment operator.

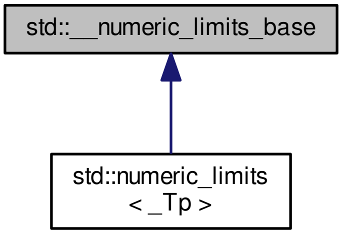
Definition at line 1781 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.509 std::__numeric_limits_base Struct Reference

Inheritance diagram for std::__numeric_limits_base:



Static Public Attributes

- static constexpr int digits
- static constexpr int digits10

- static constexpr float_denorm_style has_denorm
- static constexpr bool has_denorm_loss
- static constexpr bool has_infinity
- static constexpr bool has_quiet_NaN
- static constexpr bool has_signaling_NaN
- static constexpr bool is_bounded
- static constexpr bool is_exact
- static constexpr bool is_iec559
- static constexpr bool is_integer
- static constexpr bool is_modulo
- static constexpr bool is_signed
- static constexpr bool is_specialized
- static constexpr int max_digits10
- static constexpr int max_exponent
- static constexpr int max_exponent10
- static constexpr int min_exponent
- static constexpr int min_exponent10
- static constexpr int radix
- static constexpr float_round_style round_style
- static constexpr bool tinyness_before
- static constexpr bool traps

4.509.1 Detailed Description

Part of std::numeric_limits.

The `static const` members are usable as integral constant expressions.

Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the std::numeric_limits class.

Definition at line 191 of file limits.

4.509.2 Member Data Documentation

4.509.2.1 constexpr int std::__numeric_limits_base::digits `[static]`

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 200 of file limits.

4.509.2.2 constexpr int std::__numeric_limits_base::digits10 `[static]`

The number of base 10 digits that can be represented without change.

Definition at line 203 of file limits.

4.509.2.3 constexpr float_denorm_style std::__numeric_limits_base::has_denorm `[static]`

See std::float_denorm_style for more information.

Definition at line 255 of file limits.

4.509.2.4 constexpr bool std::__numeric_limits_base::has_denorm_loss [static]

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 259 of file limits.

4.509.2.5 constexpr bool std::__numeric_limits_base::has_infinity [static]

True if the type has a representation for positive infinity.

Definition at line 244 of file limits.

4.509.2.6 constexpr bool std::__numeric_limits_base::has_quiet_NaN [static]

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 248 of file limits.

4.509.2.7 constexpr bool std::__numeric_limits_base::has_signaling_NaN [static]

True if the type has a representation for a signaling Not a Number.

Definition at line 252 of file limits.

4.509.2.8 constexpr bool std::__numeric_limits_base::is_bounded [static]

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 268 of file limits.

4.509.2.9 constexpr bool std::__numeric_limits_base::is_exact [static]

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 220 of file limits.

4.509.2.10 constexpr bool std::__numeric_limits_base::is_iec559 [static]

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 263 of file limits.

4.509.2.11 constexpr bool std::__numeric_limits_base::is_integer [static]

True if the type is integer.

Definition at line 215 of file limits.

4.509.2.12 constexpr bool std::__numeric_limits_base::is_modulo [static]

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 277 of file limits.

4.509.2.13 constexpr bool std::__numeric_limits_base::is_signed [static]

True if the type is signed.

Definition at line 212 of file limits.

4.509.2.14 constexpr bool std::__numeric_limits_base::is_specialized [static]

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 195 of file limits.

4.509.2.15 constexpr int std::__numeric_limits_base::max_digits10 [static]

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 208 of file limits.

4.509.2.16 constexpr int std::__numeric_limits_base::max_exponent [static]

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 237 of file limits.

4.509.2.17 constexpr int std::__numeric_limits_base::max_exponent10 [static]

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 241 of file limits.

4.509.2.18 constexpr int std::__numeric_limits_base::min_exponent [static]

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 228 of file limits.

4.509.2.19 constexpr int std::__numeric_limits_base::min_exponent10 [static]

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 232 of file limits.

4.509.2.20 constexpr int std::__numeric_limits_base::radix [static]

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 224 of file limits.

4.509.2.21 constexpr float_round_style std::__numeric_limits_base::round_style [static]

See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

Definition at line 288 of file limits.

4.509.2.22 constexpr bool std::__numeric_limits_base::tinyness_before [static]

True if tininess is detected before rounding. (see IEC 559)

Definition at line 283 of file limits.

4.509.2.23 constexpr bool std::__numeric_limits_base::traps [static]

True if trapping is implemented for this type.

Definition at line 280 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.510 std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference

Public Member Functions

- int **operator()** (int __limit)

4.510.1 Detailed Description

template<typename _MustBeInt = int>struct std::__parallel::_CRandNumber< _MustBeInt >

Functor wrapper for std::rand().

Definition at line 1656 of file algo.h.

The documentation for this struct was generated from the following file:

- algo.h

4.511 std::__profile::bitset< _Nb > Class Template Reference

Inherits bitset< _Nb >.

Public Member Functions

- constexpr **bitset** (unsigned long long __val) noexcept
- template<typename _CharT , typename _Traits , typename _Alloc >
  bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __pos=0, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n=(std::basic_string< _CharT, _Traits, _Alloc >::npos))
- template<class _CharT , class _Traits , class _Alloc >
  bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __pos, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))
- **bitset** (const _Base &__x)
- template<typename _CharT >
  bitset (const _CharT ∗__str, typename std::basic_string< _CharT >::size_type __n=std::basic_string< _CharT >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))

- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- bitset< _Nb > & **flip** () noexcept
- bitset< _Nb > & **flip** (size_t __pos)
- bool **operator!=** (const bitset< _Nb > &__rhs) const noexcept
- bitset< _Nb > & **operator&=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > **operator<<** (size_t __pos) const noexcept
- bitset< _Nb > & **operator<<=** (size_t __pos) noexcept
- bool **operator==** (const bitset< _Nb > &__rhs) const noexcept
- bitset< _Nb > **operator>>** (size_t __pos) const noexcept
- bitset< _Nb > & **operator>>=** (size_t __pos) noexcept
- reference **operator[]** (size_t __pos)
- constexpr bool **operator[]** (size_t __pos) const
- bitset< _Nb > & **operator^=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > & **operator|=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > **operator~** () const noexcept
- bitset< _Nb > & **reset** () noexcept
- bitset< _Nb > & **reset** (size_t __pos)
- bitset< _Nb > & **set** () noexcept
- bitset< _Nb > & **set** (size_t __pos, bool __val=true)
- template<typename _CharT , typename _Traits , typename _Alloc >
  std::basic_string< _CharT,
  _Traits, _Alloc > **to_string** () const
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_string< _CharT,
  _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<typename _CharT , typename _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **to_string** () const
- template<class _CharT , class _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<typename _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **to_string** () const
- template<class _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- std::basic_string< char,
  std::char_traits< char >
  , std::allocator< char > > **to_string** () const
- std::basic_string< char,
  std::char_traits< char >
  , std::allocator< char > > **to_string** (char __zero, char __one= '1') const

4.511.1 Detailed Description

template<size_t _Nb>class std::__profile::bitset< _Nb >

Class std::bitset wrapper with performance instrumentation.

Definition at line 40 of file profile/bitset.

The documentation for this class was generated from the following file:

- profile/bitset

4.512 std::__profile::deque< _Tp, _Allocator > Class Template Reference

Inherits deque< _Tp, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::reverse_iterator **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **deque** (const _Allocator &__a=_Allocator())
- **deque** (size_type __n)
- **deque** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> **deque** (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **deque** (const deque &__x)
- **deque** (const _Base &__x)
- **deque** (deque &&__x)
- **deque** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- void **assign** (initializer_list< value_type > __l)
- reference **back** ()
- const_reference **back** () const

- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (iterator __position, _Args &&...__args)
- template<typename... _Args>
  void **emplace_back** (_Args &&...__args)
- template<typename... _Args>
  void **emplace_front** (_Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __p, initializer_list< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- deque & **operator=** (const deque &__x)
- deque & **operator=** (deque &&__x)
- deque & **operator=** (initializer_list< value_type > __l)
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (const _Tp &__x)
- void **push_back** (_Tp &&__x)
- void **push_front** (const _Tp &__x)
- void **push_front** (_Tp &&__x)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **swap** (deque &__x)

4.512.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__profile::deque< _Tp, _Allocator >

Class std::deque wrapper with performance instrumentation.

Definition at line 40 of file profile/deque.

The documentation for this class was generated from the following file:

- profile/deque

4.513 std::__profile::forward_list< _Tp, _Alloc > Class Template Reference

Inherits forward_list< _Tp, _Alloc >.

Public Types

- typedef _Base::size_type **size_type**

Public Member Functions

- **forward_list** (const _Alloc &__al=_Alloc())
- **forward_list** (const forward_list &__list, const _Alloc &__al)
- **forward_list** (forward_list &&__list, const _Alloc &__al)
- **forward_list** (size_type __n, const _Alloc &__al=_Alloc())
- **forward_list** (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- **forward_list** (const forward_list &__list)
- **forward_list** (forward_list &&__list) noexcept
- **forward_list** (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- forward_list & **operator=** (const forward_list &__list)
- forward_list & **operator=** (forward_list &&__list) noexcept(_Node_alloc_traits
- forward_list & **operator=** (std::initializer_list< _Tp > __il)

4.513.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>class std::__profile::forward_list< _Tp, _Alloc >

Class std::forward_list wrapper with performance instrumentation.

Definition at line 44 of file profile/forward_list.

The documentation for this class was generated from the following file:

- profile/forward_list

4.514 std::__profile::list< _Tp, _Allocator > Class Template Reference

Inherits list< _Tp, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef __iterator_tracker
  < typename
  _Base::const_iterator, list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator
  < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __iterator_tracker
  < typename _Base::iterator,
  list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **list** (const _Allocator &__a=_Allocator())
- **list** (size_type __n)
- **list** (size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** (const list &__x)
- **list** (const _Base &__x)
- **list** (list &&__x) noexcept
- **list** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void **_M_profile_find** () const
- void **_M_profile_iterate** (int __rewind=0) const
- void **assign** (initializer_list< value_type > __l)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__t)
- reference **back** ()
- const_reference **back** () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept

- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (iterator __position, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __position, iterator __last)
- reference **front** ()
- const_reference **front** () const
- iterator **insert** (iterator __position, const _Tp &__x)
- iterator **insert** (iterator __position, _Tp &&__x)
- void **insert** (iterator __position, initializer_list< value_type > __l)
- void **insert** (iterator __position, size_type __n, const _Tp &__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- void **merge** (list &&__x)
- void **merge** (list &__x)
- template<class _Compare >
  void **merge** (list &&__x, _Compare __comp)
- template<typename _Compare >
  void **merge** (list &__x, _Compare __comp)
- list & **operator=** (const list &__x)
- list & **operator=** (list &&__x)
- list & **operator=** (initializer_list< value_type > __l)
- void **pop_back** ()
- void **pop_front** ()
- void **push_front** (const value_type &__x)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- void **remove** (const _Tp &__value)
- template<class _Predicate >
  void **remove_if** (_Predicate __pred)
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **sort** ()
- template<typename _StrictWeakOrdering >
  void **sort** (_StrictWeakOrdering __pred)
- void **splice** (iterator __position, list &&__x)
- void **splice** (iterator __position, list &__x)
- void **splice** (iterator __position, list &__x, iterator __i)
- void **splice** (iterator __position, list &&__x, iterator __i)
- void **splice** (iterator __position, list &&__x, iterator __first, iterator __last)
- void **splice** (iterator __position, list &__x, iterator __first, iterator __last)
- void **swap** (list &__x)
- void **unique** ()
- template<class _BinaryPredicate >
  void **unique** (_BinaryPredicate __binary_pred)

4.514.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>class std::__profile::list< _Tp, _Allocator >

List wrapper with performance instrumentation.

Definition at line 42 of file profile/list.

The documentation for this class was generated from the following file:

- profile/list

4.515 std::__profile::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference

Inherits map< _Key, _Tp, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator
  < const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key,
  _Tp > **value_type**

Public Member Functions

- **map** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **map** (const map &__x)
- **map** (const _Base &__x)
- **map** (map &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **map** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- mapped_type & **at** (const key_type &__k)

- const mapped_type & **at** (const key_type &__k) const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- size_type **count** (const key_type &__x) const
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- std::pair< iterator, bool > **insert** (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__x)
- void **insert** (std::initializer_list< value_type > __list)
- iterator **insert** (const_iterator __position, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- map & **operator=** (const map &__x)
- map & **operator=** (map &&__x)
- map & **operator=** (initializer_list< value_type > __l)
- mapped_type & **operator[ ]** (const key_type &__k)
- mapped_type & **operator[ ]** (key_type &&__k)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (map &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.515.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>class std::__profile::map< _Key, _Tp, _Compare, _Allocator >

Class std::map wrapper with performance instrumentation.

Definition at line 41 of file profile/map.h.

The documentation for this class was generated from the following file:

- profile/map.h

4.516 std::__profile::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference

Inherits multimap< _Key, _Tp, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::reverse_iterator **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- **multimap** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> **multimap** (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multimap** (const multimap &__x)
- **multimap** (const _Base &__x)
- **multimap** (multimap &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **multimap** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- iterator **begin** () noexcept

- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__x)
- void **insert** (std::initializer_list< value_type > __list)
- iterator **insert** (const_iterator __position, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multimap & **operator=** (const multimap &__x)
- multimap & **operator=** (multimap &&__x)
- multimap & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (multimap &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.516.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >

Class std::multimap wrapper with performance instrumentation.

Definition at line 41 of file profile/multimap.h.

The documentation for this class was generated from the following file:

- profile/multimap.h

4.517 std::__profile::multiset< _Key, _Compare, _Allocator > Class Template Reference

Inherits multiset< _Key, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::reverse_iterator **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **multiset** (const multiset &__x)
- **multiset** (const _Base &__x)
- **multiset** (multiset &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **multiset** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept

- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- iterator **insert** (const value_type &__x)
- iterator **insert** (value_type &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list< value_type > __l)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multiset & **operator=** (const multiset &__x)
- multiset & **operator=** (multiset &&__x)
- multiset & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (multiset &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.517.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>class std::__profile::multiset< _Key, _Compare, _Allocator >

Class std::multiset wrapper with performance instrumentation.

Definition at line 41 of file profile/multiset.h.

The documentation for this class was generated from the following file:

- profile/multiset.h

4.518 std::__profile::set< _Key, _Compare, _Allocator > Class Template Reference

Inherits set< _Key, _Compare, _Allocator >.

Public Types

- typedef _Allocator **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::const_reverse_iterator **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::reverse_iterator **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **set** (const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> **set** (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- **set** (const set &__x)
- **set** (const _Base &__x)
- **set** (set &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- **set** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args> std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args> iterator **emplace_hint** (const_iterator __pos, _Args &&...__args)
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- iterator **erase** (const_iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)

- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- std::pair< iterator, bool > **insert** (const value_type &__x)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void **insert** (_InputIterator __first, _InputIterator __last)
- void **insert** (initializer_list< value_type > __l)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- set & **operator=** (const set &__x)
- set & **operator=** (set &&__x)
- set & **operator=** (initializer_list< value_type > __l)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **swap** (set &__x)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

4.518.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>class std::__profile::set< _Key, _Compare, _Allocator >

Class std::set wrapper with performance instrumentation.

Definition at line 41 of file profile/set.h.

The documentation for this class was generated from the following file:

- profile/set.h

4.519 std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Inherits unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef _Base::mapped_type **mapped_type**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_map** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template< typename _InputIterator >
  unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const unordered_map &__x)
- **unordered_map** (const _Base &__x)
- **unordered_map** (unordered_map &&__x)
- **unordered_map** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- _Base & **_M_base** () noexcept
- const _Base & **_M_base** () const noexcept
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __it, _Args &&...__args)
- void **insert** (std::initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__obj)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __iter, _Pair &&__v)
- template<typename _InputIter >
  void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type ∗__first, const value_type ∗__last)
- unordered_map & **operator=** (const unordered_map &__x)
- unordered_map & **operator=** (unordered_map &&__x)
- unordered_map & **operator=** (initializer_list< value_type > __l)
- mapped_type & **operator[ ]** (const _Key &__k)
- mapped_type & **operator[ ]** (_Key &&__k)
- void **rehash** (size_type __n)
- void **swap** (unordered_map &__x)

4.519.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_map wrapper with performance instrumentation.

Definition at line 50 of file profile/unordered_map.

The documentation for this class was generated from the following file:

- profile/unordered_map

4.520 std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Inherits unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multimap** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_multimap (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (const unordered_multimap &__x)
- **unordered_multimap** (const _Base &__x)
- **unordered_multimap** (unordered_multimap &&__x)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __it, _Args &&...__args)
- void **insert** (std::initializer_list< value_type > __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__obj)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __iter, _Pair &&__v)
- template<typename _InputIter >
  void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type $*$__first, const value_type $*$__last)
- unordered_multimap & **operator=** (const unordered_multimap &__x)
- unordered_multimap & **operator=** (unordered_multimap &&__x)
- unordered_multimap & **operator=** (initializer_list< value_type > __l)
- void **rehash** (size_type __n)
- void **swap** (unordered_multimap &__x)

4.520.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash< _Key >, typename _Pred = std::equal_to< _Key >, typename _Alloc = std::allocator< _Key > >class std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_multimap wrapper with performance instrumentation.

Definition at line 352 of file profile/unordered_map.

The documentation for this class was generated from the following file:

- profile/unordered_map

4.521 std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

Inherits unordered_multiset< _Value, _Hash, _Pred, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multiset** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_multiset (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const unordered_multiset &__x)
- **unordered_multiset** (const _Base &__x)
- **unordered_multiset** (unordered_multiset &&__x)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __it, _Args &&...__args)
- void **insert** (std::initializer_list< value_type > __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)

- iterator **insert** (value_type &&__obj)
- iterator **insert** (const_iterator __iter, value_type &&__v)
- template<typename _InputIter >
  void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type ∗__first, const value_type ∗__last)
- unordered_multiset & **operator=** (const unordered_multiset &__x)
- unordered_multiset & **operator=** (unordered_multiset &&__x)
- unordered_multiset & **operator=** (initializer_list< value_type > __l)
- void **rehash** (size_type __n)
- void **swap** (unordered_multiset &__x)

4.521.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

Unordered_multiset wrapper with performance instrumentation.

Definition at line 318 of file profile/unordered_set.

The documentation for this class was generated from the following file:

- profile/unordered_set

4.522 std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc > Class Template Reference

Inherits unordered_set< _Key, _Hash, _Pred, _Alloc >.

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef _Base::iterator **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_set (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (const unordered_set &__x)

- **unordered_set** (const _Base &__x)
- **unordered_set** (unordered_set &&__x)
- **unordered_set** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __it, _Args &&...__args)
- void **insert** (std::initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __iter, const value_type &__v)
- std::pair< iterator, bool > **insert** (value_type &&__obj)
- iterator **insert** (const_iterator __iter, value_type &&__v)
- template<typename _InputIter >
  void **insert** (_InputIter __first, _InputIter __last)
- void **insert** (const value_type ∗__first, const value_type ∗__last)
- unordered_set & **operator=** (const unordered_set &__x)
- unordered_set & **operator=** (unordered_set &&__x)
- unordered_set & **operator=** (initializer_list< value_type > __l)
- void **rehash** (size_type __n)
- void **swap** (unordered_set &__x)

4.522.1 Detailed Description

template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<_Key>>class std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >

Unordered_set wrapper with performance instrumentation.

Definition at line 50 of file profile/unordered_set.

The documentation for this class was generated from the following file:

- profile/unordered_set

4.523 std::_Base_bitset< _Nw > Struct Template Reference

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- template<size_t _Nb>
  bool **_M_are_all** () const noexcept
- void **_M_do_and** (const _Base_bitset< _Nw > &__x) noexcept
- size_t **_M_do_count** () const noexcept
- size_t **_M_do_find_first** (size_t) const noexcept
- size_t **_M_do_find_next** (size_t, size_t) const noexcept

- void **_M_do_flip** () noexcept
- void **_M_do_left_shift** (size_t __shift) noexcept
- void **_M_do_or** (const _Base_bitset< _Nw > &__x) noexcept
- void **_M_do_reset** () noexcept
- void **_M_do_right_shift** (size_t __shift) noexcept
- void **_M_do_set** () noexcept
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const _Base_bitset< _Nw > &__x) noexcept
- const _WordT ∗ **_M_getdata** () const noexcept
- _WordT & **_M_getword** (size_t __pos) noexcept
- constexpr _WordT **_M_getword** (size_t __pos) const noexcept
- _WordT & **_M_hiword** () noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- bool **_M_is_any** () const noexcept
- bool **_M_is_equal** (const _Base_bitset< _Nw > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT _M_w [_Nw]

4.523.1 Detailed Description

template<size_t _Nw>struct std::_Base_bitset< _Nw >

Base class, general case. It is a class invariant that _Nw will be nonnegative.

See documentation for bitset.

Definition at line 71 of file bitset.

4.523.2 Member Data Documentation

4.523.2.1 template<size_t _Nw> _WordT std::_Base_bitset< _Nw >::_M_w[_Nw]

0 is the least significant word.

Definition at line 76 of file bitset.

The documentation for this struct was generated from the following file:

- bitset

4.524 std::_Base_bitset< 0 > Struct Template Reference

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long) noexcept
- template<size_t _Nb>
  bool **_M_are_all** () const noexcept
- void **_M_do_and** (const _Base_bitset< 0 > &) noexcept
- size_t **_M_do_count** () const noexcept
- size_t **_M_do_find_first** (size_t) const noexcept
- size_t **_M_do_find_next** (size_t, size_t) const noexcept
- void **_M_do_flip** () noexcept
- void **_M_do_left_shift** (size_t) noexcept
- void **_M_do_or** (const _Base_bitset< 0 > &) noexcept
- void **_M_do_reset** () noexcept
- void **_M_do_right_shift** (size_t) noexcept
- void **_M_do_set** () noexcept
- unsigned long long **_M_do_to_ullong** () const noexcept
- unsigned long **_M_do_to_ulong** () const noexcept
- void **_M_do_xor** (const _Base_bitset< 0 > &) noexcept
- _WordT & **_M_getword** (size_t) noexcept
- constexpr _WordT **_M_getword** (size_t __pos) const noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- bool **_M_is_any** () const noexcept
- bool **_M_is_equal** (const _Base_bitset< 0 > &) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

4.524.1 Detailed Description

template<>struct std::_Base_bitset< 0 >

Base class, specialization for no storage (zero-length bitset).

See documentation for bitset.

Definition at line 519 of file bitset.

The documentation for this struct was generated from the following file:

- bitset

4.525 std::_Base_bitset< 1 > Struct Template Reference

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- template<size_t _Nb>
  bool **_M_are_all** () const noexcept
- void **_M_do_and** (const _Base_bitset< 1 > &__x) noexcept
- size_t **_M_do_count** () const noexcept
- size_t **_M_do_find_first** (size_t __not_found) const noexcept
- size_t **_M_do_find_next** (size_t __prev, size_t __not_found) const noexcept
- void **_M_do_flip** () noexcept
- void **_M_do_left_shift** (size_t __shift) noexcept
- void **_M_do_or** (const _Base_bitset< 1 > &__x) noexcept
- void **_M_do_reset** () noexcept
- void **_M_do_right_shift** (size_t __shift) noexcept
- void **_M_do_set** () noexcept
- unsigned long long **_M_do_to_ullong** () const noexcept
- unsigned long **_M_do_to_ulong** () const noexcept
- void **_M_do_xor** (const _Base_bitset< 1 > &__x) noexcept
- const _WordT ∗ **_M_getdata** () const noexcept
- _WordT & **_M_getword** (size_t) noexcept
- constexpr _WordT **_M_getword** (size_t) const noexcept
- _WordT & **_M_hiword** () noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- bool **_M_is_any** () const noexcept
- bool **_M_is_equal** (const _Base_bitset< 1 > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT **_M_w**

4.525.1 Detailed Description

template<>struct std::_Base_bitset< 1 >

Base class, specialization for a single word.

See documentation for bitset.

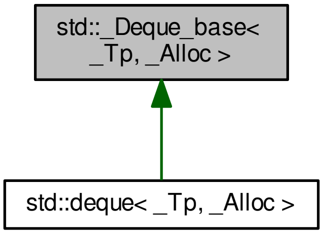
Definition at line 372 of file bitset.

The documentation for this struct was generated from the following file:

- bitset

4.526 std::_Deque_base< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::_Deque_base< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Deque_iterator< _Tp, const _Tp &, const _Tp * > **const_iterator**
- typedef _Deque_iterator< _Tp, _Tp &, _Tp * > **iterator**

Public Member Functions

- **_Deque_base** (size_t __num_elements)
- **_Deque_base** (const allocator_type &__a, size_t __num_elements)
- **_Deque_base** (const allocator_type &__a)
- **_Deque_base** (_Deque_base &&__x)
- allocator_type **get_allocator** () const noexcept

Protected Types

- enum { **_S_initial_map_size** }
- typedef _Alloc::template rebind< _Tp * >::other **_Map_alloc_type**
- typedef _Alloc::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- _Tp ** **_M_allocate_map** (size_t __n)
- _Tp * **_M_allocate_node** ()
- void **_M_create_nodes** (_Tp **__nstart, _Tp **__nfinish)
- void **_M_deallocate_map** (_Tp **__p, size_t __n)
- void **_M_deallocate_node** (_Tp *__p)
- void **_M_destroy_nodes** (_Tp **__nstart, _Tp **__nfinish)
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- void _M_initialize_map (size_t)

Protected Attributes

- _Deque_impl **_M_impl**

4.526.1 Detailed Description

template<typename _Tp, typename _Alloc>class std::_Deque_base< _Tp, _Alloc >

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 439 of file stl_deque.h.

4.526.2 Member Function Documentation

4.526.2.1 template<typename _Tp , typename _Alloc > void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t *__num_elements*) `[protected]`

Layout storage.

Parameters

__num_elements	The count of T's for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 582 of file stl_deque.h.

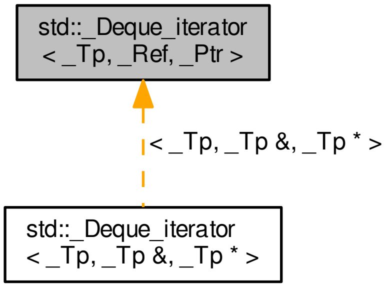
References std::max().

The documentation for this class was generated from the following file:

- stl_deque.h

4.527 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

Inheritance diagram for std::_Deque_iterator< _Tp, _Ref, _Ptr >:



Public Types

- typedef _Tp ∗∗ **_Map_pointer**
- typedef _Deque_iterator **_Self**
- typedef _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > **const_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **iterator**
- typedef std::random_access_iterator_tag **iterator_category**
- typedef _Ptr **pointer**
- typedef _Ref **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_Deque_iterator** (_Tp ∗__x, _Map_pointer __y)
- **_Deque_iterator** (const iterator &__x)
- void _M_set_node (_Map_pointer __new_node)
- reference **operator∗** () const
- _Self **operator+** (difference_type __n) const
- _Self & **operator++** ()
- _Self **operator++** (int)
- _Self & **operator+=** (difference_type __n)
- _Self **operator-** (difference_type __n) const
- _Self & **operator--** ()
- _Self **operator--** (int)
- _Self & **operator-=** (difference_type __n)
- pointer **operator->** () const
- reference **operator[ ]** (difference_type __n) const

Static Public Member Functions

- static size_t **_S_buffer_size** ()

Public Attributes

- _Tp ∗ **_M_cur**
- _Tp ∗ **_M_first**
- _Tp ∗ **_M_last**
- _Map_pointer **_M_node**

4.527.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr>struct std::_Deque_iterator< _Tp, _Ref, _Ptr >

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for _M_set_node.

Definition at line 106 of file stl_deque.h.

4.527.2 Member Function Documentation

4.527.2.1 template<typename _Tp, typename _Ref, typename _Ptr> void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (_Map_pointer *__new_node*) [inline]

Prepares to traverse new_node. Sets everything except _M_cur, which should therefore be set by the caller immediately afterwards, based on _M_first and _M_last.

Definition at line 234 of file stl_deque.h.

The documentation for this struct was generated from the following file:

- stl_deque.h

4.528 std::_Derives_from_binary_function< _Tp > Struct Template Reference

Inherits std::__sfinae_types.

Public Types

- typedef char **__one**

Static Public Attributes

- static const bool **value**

4.528.1 Detailed Description

template<typename _Tp>struct std::_Derives_from_binary_function< _Tp >

Determines if the type _Tp derives from binary_function.

Definition at line 206 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.529 std::_Derives_from_unary_function< _Tp > Struct Template Reference

Inherits std::__sfinae_types.

Public Types

- typedef char **__one**

Static Public Attributes

- static const bool **value**

4.529.1 Detailed Description

template<typename _Tp>struct std::_Derives_from_unary_function< _Tp >

Determines if the type _Tp derives from unary_function.

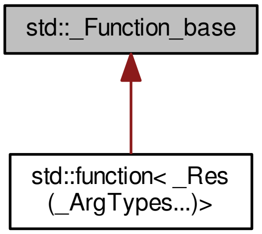
Definition at line 190 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.530 std::_Function_base Class Reference

Inheritance diagram for std::_Function_base:



Public Types

- typedef bool(∗ **_Manager_type**)(_Any_data &, const _Any_data &, _Manager_operation)

Public Member Functions

- bool **_M_empty** () const

Public Attributes

- _Any_data **_M_functor**
- _Manager_type **_M_manager**

Static Public Attributes

- static const std::size_t **_M_max_align**
- static const std::size_t **_M_max_size**

4.530.1 Detailed Description

Base class of all polymorphic function object wrappers.

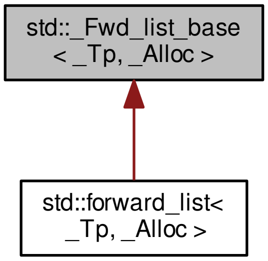
Definition at line 1869 of file functional.

The documentation for this class was generated from the following file:

- functional

4.531 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference

Inheritance diagram for std::_Fwd_list_base< _Tp, _Alloc >:



Public Types

- typedef _Fwd_list_node< _Tp > **_Node**
- typedef _Fwd_list_const_iterator< _Tp > **const_iterator**
- typedef _Fwd_list_iterator< _Tp > **iterator**

Public Member Functions

- **_Fwd_list_base** (const _Node_alloc_type &__a)
- **_Fwd_list_base** (_Fwd_list_base &&__lst, const _Node_alloc_type &__a)
- **_Fwd_list_base** (_Fwd_list_base &&__lst)
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept

Protected Types

- typedef __gnu_cxx::__alloc_traits< _Alloc > **_Alloc_traits**
- typedef __gnu_cxx::__alloc_traits< _Node_alloc_type > **_Node_alloc_traits**
- typedef _Alloc_traits::template rebind< _Fwd_list_node< _Tp > >::other **_Node_alloc_type**
- typedef _Alloc_traits::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- template<typename... _Args>
  _Node ∗ **_M_create_node** (_Args &&...__args)
- _Fwd_list_node_base ∗ **_M_erase_after** (_Fwd_list_node_base ∗__pos)
- _Fwd_list_node_base ∗ **_M_erase_after** (_Fwd_list_node_base ∗__pos, _Fwd_list_node_base ∗__last)
- _Node ∗ **_M_get_node** ()
- template<typename... _Args>
  _Fwd_list_node_base ∗ **_M_insert_after** (const_iterator __pos, _Args &&...__args)
- void **_M_put_node** (_Node ∗__p)

Protected Attributes

- _Fwd_list_impl **_M_impl**

4.531.1 Detailed Description

template<typename _Tp, typename _Alloc>struct std::_Fwd_list_base< _Tp, _Alloc >

Base class for forward_list.

Definition at line 275 of file forward_list.h.

The documentation for this struct was generated from the following files:

- forward_list.h
- forward_list.tcc

4.532 std::_Fwd_list_const_iterator< _Tp > Struct Template Reference

Public Types

- typedef const _Fwd_list_node< _Tp > **_Node**
- typedef
  _Fwd_list_const_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef _Fwd_list_iterator< _Tp > **iterator**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const _Tp ∗ **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_Fwd_list_const_iterator** (const _Fwd_list_node_base ∗__n)
- **_Fwd_list_const_iterator** (const iterator &__iter)
- _Self **_M_next** () const
- bool **operator!=** (const _Self &__x) const
- reference **operator**∗ () const
- _Self & **operator++** ()
- _Self **operator++** (int)
- pointer **operator->** () const
- bool **operator==** (const _Self &__x) const

Public Attributes

- const _Fwd_list_node_base ∗ **_M_node**

4.532.1 Detailed Description

template<typename _Tp>struct std::_Fwd_list_const_iterator< _Tp >

A forward_list::const_iterator.

All the functions are op overloads.

Definition at line 188 of file forward_list.h.

The documentation for this struct was generated from the following file:

- forward_list.h

4.533 std::_Fwd_list_iterator< _Tp > Struct Template Reference

Public Types

- typedef _Fwd_list_node< _Tp > **_Node**
- typedef _Fwd_list_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef _Tp ∗ **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_Fwd_list_iterator** (_Fwd_list_node_base ∗__n)
- _Self **_M_next** () const
- bool **operator!=** (const _Self &__x) const
- reference **operator∗** () const
- _Self & **operator++** ()
- _Self **operator++** (int)
- pointer **operator->** () const
- bool **operator==** (const _Self &__x) const

Public Attributes

- _Fwd_list_node_base ∗ **_M_node**

4.533.1 Detailed Description

template<typename _Tp>struct std::_Fwd_list_iterator< _Tp >

A forward_list::iterator.

All the functions are op overloads.

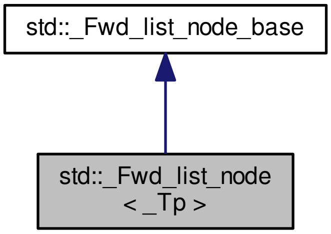
Definition at line 121 of file forward_list.h.

The documentation for this struct was generated from the following file:

- forward_list.h

4.534 std::_Fwd_list_node< _Tp > Struct Template Reference

Inheritance diagram for std::_Fwd_list_node< _Tp >:



Public Member Functions

- void **_M_reverse_after** () noexcept
- _Fwd_list_node_base ∗ **_M_transfer_after** (_Fwd_list_node_base ∗__begin, _Fwd_list_node_base ∗__end)
- _Tp ∗ **_M_valptr** () noexcept
- const _Tp ∗ **_M_valptr** () const noexcept

Public Attributes

- _Fwd_list_node_base ∗ **_M_next**
- aligned_storage< sizeof(_Tp),
  alignment_of< _Tp >::value >
  ::type **_M_storage**

4.534.1 Detailed Description

template<typename _Tp>struct std::_Fwd_list_node< _Tp >

A helper node class for forward_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

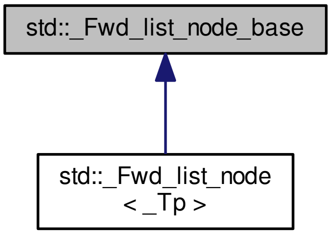
Definition at line 94 of file forward_list.h.

The documentation for this struct was generated from the following file:

- forward_list.h

4.535 std::_Fwd_list_node_base Struct Reference

Inheritance diagram for std::_Fwd_list_node_base:



Public Member Functions

- void **_M_reverse_after** () noexcept
- _Fwd_list_node_base ∗ **_M_transfer_after** (_Fwd_list_node_base ∗__begin, _Fwd_list_node_base ∗__end)

Public Attributes

- _Fwd_list_node_base ∗ **_M_next**

4.535.1 Detailed Description

A helper basic node class for forward_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

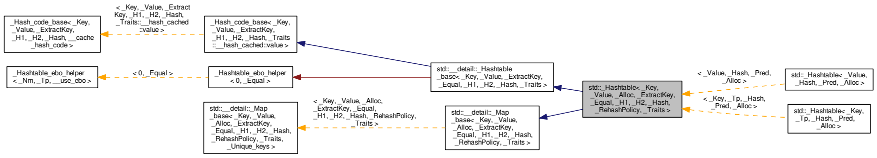
Definition at line 49 of file forward_list.h.

The documentation for this struct was generated from the following file:

- forward_list.h

4.536 std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Class Template Reference

Inheritance diagram for std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >:



Public Types

- typedef _Alloc **allocator_type**
- using **const_iterator** = typename __hashtable_base::const_iterator
- using **const_local_iterator** = typename __hashtable_base::const_local_iterator
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- using **difference_type** = typename __hashtable_base::difference_type
- using **iterator** = typename __hashtable_base::iterator
- typedef _Equal **key_equal**
- typedef _Key **key_type**
- using **local_iterator** = typename __hashtable_base::local_iterator
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- using **size_type** = typename __hashtable_base::size_type
- typedef _Value **value_type**

Public Member Functions

- **_Hashtable** (size_type __bucket_hint, const _H1 &, const _H2 &, const _Hash &, const _Equal &, const _ExtractKey &, const allocator_type &)
- template< typename _InputIterator >
  _Hashtable (_InputIterator __first, _InputIterator __last, size_type __bucket_hint, const _H1 &, const _H2 &, const _Hash &, const _Equal &, const _ExtractKey &, const allocator_type &)
- **_Hashtable** (const _Hashtable &)
- **_Hashtable** (_Hashtable &&)
- **_Hashtable** (size_type __n=10, const _H1 &__hf=_H1(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template< typename _InputIterator >
  _Hashtable (_InputIterator __f, _InputIterator __l, size_type __n=0, const _H1 &__hf=_H1(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **_Hashtable** (initializer_list< value_type > __l, size_type __n=0, const _H1 &__hf=_H1(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- const _RehashPolicy & **__rehash_policy** () const
- void **__rehash_policy** (const _RehashPolicy &)
- iterator **begin** () noexcept

- const_iterator **begin** () const noexcept
- local_iterator **begin** (size_type __n)
- const_local_iterator **begin** (size_type __n) const
- size_type **bucket** (const key_type &__k) const
- size_type **bucket_count** () const noexcept
- size_type **bucket_size** (size_type __n) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __n) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __n) const
- void **clear** () noexcept
- size_type **count** (const key_type &__k) const
- template<typename... _Args>
  __ireturn_type **emplace** (_Args &&...__args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator, _Args &&...__args)
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- local_iterator **end** (size_type __n)
- const_local_iterator **end** (size_type __n) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__k)
- std::pair< const_iterator,
  const_iterator > **equal_range** (const key_type &__k) const
- iterator **erase** (const_iterator)
- iterator **erase** (iterator __it)
- size_type **erase** (const key_type &__k)
- iterator **erase** (const_iterator, const_iterator)
- iterator **find** (const key_type &__k)
- const_iterator **find** (const key_type &__k) const
- allocator_type **get_allocator** () const noexcept
- key_equal **key_eq** () const
- float **load_factor** () const noexcept
- size_type **max_bucket_count** () const noexcept
- size_type **max_size** () const noexcept
- _Hashtable & **operator=** (const _Hashtable &__ht)
- _Hashtable & **operator=** (_Hashtable &&__ht)
- _Hashtable & **operator=** (initializer_list< value_type > __l)
- void **rehash** (size_type __n)
- size_type **size** () const noexcept
- void **swap** (_Hashtable &)

Protected Member Functions

- size_type **_M_bucket_index** (__node_type *__n) const
- size_type **_M_bucket_index** (const key_type &__k, __hash_code __c) const
- template<typename... _Args>
  std::pair< iterator, bool > **_M_emplace** (std::true_type, _Args &&...__args)
- template<typename... _Args>
  iterator **_M_emplace** (std::false_type, _Args &&...__args)
- const _Equal & **_M_eq** () const

- _Equal & **_M_eq** ()
- bool **_M_equals** (const _Key &__k, __hash_code __c, __node_type ∗__n) const
- size_type **_M_erase** (std::true_type, const key_type &)
- size_type **_M_erase** (std::false_type, const key_type &)
- iterator **_M_erase** (size_type __bkt, __node_base ∗__prev_n, __node_type ∗__n)
- __node_base ∗ **_M_find_before_node** (size_type, const key_type &, __hash_code) const
- __node_type ∗ **_M_find_node** (size_type __bkt, const key_type &__key, __hash_code __c) const
- __node_base ∗ **_M_get_previous_node** (size_type __bkt, __node_base ∗__n)
- template<typename _Arg >
  std::pair< iterator, bool > **_M_insert** (_Arg &&, std::true_type)
- template<typename _Arg >
  iterator **_M_insert** (_Arg &&, std::false_type)
- void **_M_insert_bucket_begin** (size_type, __node_type ∗)
- iterator **_M_insert_multi_node** (__hash_code __code, __node_type ∗__n)
- iterator **_M_insert_unique_node** (size_type __bkt, __hash_code __code, __node_type ∗__n)
- void **_M_remove_bucket_begin** (size_type __bkt, __node_type ∗__next_n, size_type __next_bkt)
- void **_M_swap** (_Hashtable_base &__x)

Friends

- template<typename _Keya , typename _Valuea , typename _Alloca , typename _ExtractKeya , typename _Equala , typename _H1a , typename _H2a , typename _Hasha , typename _RehashPolicya , typename _Traitsa , bool _Constant_iteratorsa, bool _Unique_keysa>
  struct **__detail::_Insert**
- template<typename _Keya , typename _Valuea , typename _Alloca , typename _ExtractKeya , typename _Equala , typename _H1a , typename _H2a , typename _Hasha , typename _RehashPolicya , typename _Traitsa >
  struct **__detail::_Insert_base**
- template<typename _Keya , typename _Valuea , typename _Alloca , typename _ExtractKeya , typename _Equala , typename _H1a , typename _H2a , typename _Hasha , typename _RehashPolicya , typename _Traitsa , bool _Unique_keysa>
  struct **__detail::_Map_base**

4.536.1 Detailed Description

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2, typename _Hash, typename _RehashPolicy, typename _Traits>class std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Primary class template _Hashtable.

Template Parameters

_Value	CopyConstructible type.
_Key	CopyConstructible type.
_Alloc	An allocator type ([lib.allocator.requirements]) whose _Alloc::value_type is _Value. As a conforming extension, we allow for _Alloc::value_type != _Value.
_ExtractKey	Function object that takes an object of type _Value and returns a value of type _Key.
_Equal	Function object that takes two objects of type k and returns a bool-like value that is true if the two objects are considered equal.
_H1	The hash function. A unary function object with argument type _Key and result type size_t. Return values should be distributed over the entire range [0, numeric_limits<size_t>:max()].
_H2	The range-hashing function (in the terminology of Tavori and Dreizin). A binary function object whose argument types and result type are all size_t. Given arguments r and N, the return value is in the range [0, N).

_Hash	The ranged hash function (Tavori and Dreizin). A binary function whose argument types are _Key and size_t and whose result type is size_t. Given arguments k and N, the return value is in the range [0, N). Default: hash(k, N) = h2(h1(k), N). If _Hash is anything other than the default, _H1 and _H2 are ignored.
_RehashPolicy	Policy class with three members, all of which govern the bucket count. _M_next_bkt(n) returns a bucket count no smaller than n. _M_bkt_for_elements(n) returns a bucket count appropriate for an element count of n. _M_need_rehash(n_bkt, n_elt, n_ins) determines whether, if the current bucket count is n_bkt and the current element count is n_elt, we need to increase the bucket count. If so, returns make_pair(true, n), where n is the new bucket count. If not, returns make_pair(false, <anything>)
_Traits	Compile-time class with three boolean std::integral_constant members: __cache_hash_code, __constant_iterators, __unique_keys.

Each _Hashtable data structure has:

- _Bucket[] _M_buckets
- _Hash_node_base _M_bbegin
- size_type _M_bucket_count
- size_type _M_element_count

with _Bucket being _Hash_node∗ and _Hash_node containing:

- _Hash_node∗ _M_next
- Tp _M_value
- size_t _M_hash_code if cache_hash_code is true

In terms of Standard containers the hashtable is like the aggregation of:

- std::forward_list<_Node> containing the elements
- std::vector<std::forward_list<_Node>::iterator> representing the buckets

The non-empty buckets contain the node before the first node in the bucket. This design makes it possible to implement something like a std::forward_list::insert_after on container insertion and std::forward_list::erase_after on container erase calls. _M_before_begin is equivalent to std::forward_list::before_begin. Empty buckets contain nullptr. Note that one of the non-empty buckets contains &_M_before_begin which is not a dereferenceable node so the node pointer in a bucket shall never be dereferenced, only its next node can be.

Walking through a bucket's nodes requires a check on the hash code to see if each node is still in the bucket. Such a design assumes a quite efficient hash functor and is one of the reasons it is highly advisable to set __cache_hash_code to true.

The container iterators are simply built from nodes. This way incrementing the iterator is perfectly efficient independent of how many empty buckets there are in the container.

On insert we compute the element's hash code and use it to find the bucket index. If the element must be inserted in an empty bucket we add it at the beginning of the singly linked list and make the bucket point to _M_before_begin. The bucket that used to point to _M_before_begin, if any, is updated to point to its new before begin node.

On erase, the simple iterator design requires using the hash functor to get the index of the bucket to update. For this reason, when __cache_hash_code is set to false the hash functor must not throw and this is enforced by a static assertion.

Functionality is implemented by decomposition into base classes, where the derived _Hashtable class is used in _Map-_base, _Insert, _Rehash_base, and _Equality base classes to access the "this" pointer. _Hashtable_base is used in the base classes as a non-recursive, fully-completed-type so that detailed nested type information, such as iterator type and node type, can be used. This is similar to the "Curiously Recurring Template Pattern" (CRTP) technique, but uses a reconstructed, not explicitly passed, template pattern.

Base class templates are:

- __detail::_Hashtable_base
- __detail::_Map_base
- __detail::_Insert
- __detail::_Rehash_base
- __detail::_Equality

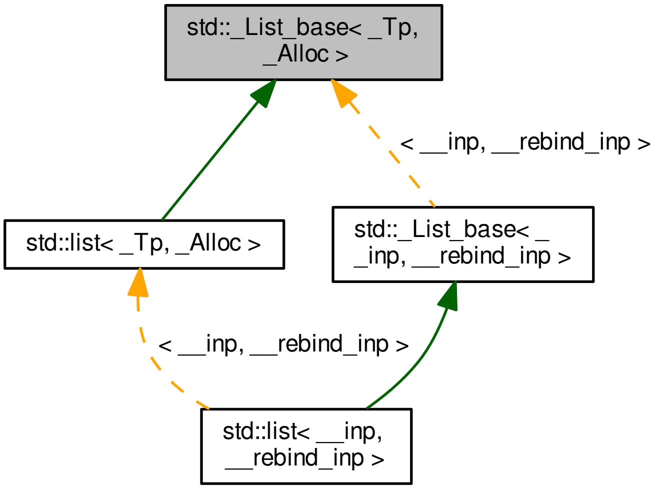
Definition at line 174 of file bits/hashtable.h.

The documentation for this class was generated from the following file:

- bits/hashtable.h

4.537 std::_List_base< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::_List_base< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**

Public Member Functions

- **_List_base** (const _Node_alloc_type &__a)
- **_List_base** (_List_base &&__x)
- void **_M_clear** ()
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Tp_alloc_type **_M_get_Tp_allocator** () const noexcept
- void **_M_init** ()
- allocator_type **get_allocator** () const noexcept

Protected Types

- typedef _Alloc::template
  rebind< _List_node< _Tp >
  >::other **_Node_alloc_type**
- typedef _Alloc::template
  rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- _List_node< _Tp > ∗ **_M_get_node** ()
- void **_M_put_node** (_List_node< _Tp > ∗__p)

Protected Attributes

- _List_impl **_M_impl**

4.537.1 Detailed Description

template<typename _Tp, typename _Alloc>class std::_List_base< _Tp, _Alloc >

See bits/stl_deque.h's _Deque_base for an explanation.

Definition at line 289 of file stl_list.h.

The documentation for this class was generated from the following files:

- stl_list.h
- list.tcc

4.538 std::_List_const_iterator< _Tp > Struct Template Reference

Public Types

- typedef const _List_node< _Tp > **_Node**
- typedef _List_const_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef _List_iterator< _Tp > **iterator**
- typedef
  std::bidirectional_iterator_tag **iterator_category**

- typedef const _Tp ∗ **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_const_iterator** (const __detail::_List_node_base ∗__x)
- **_List_const_iterator** (const iterator &__x)
- bool **operator!=** (const _Self &__x) const
- reference **operator**∗ () const
- _Self & **operator++** ()
- _Self **operator++** (int)
- _Self & **operator--** ()
- _Self **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const _Self &__x) const

Public Attributes

- const __detail::_List_node_base ∗ **_M_node**

4.538.1 Detailed Description

template<typename _Tp>struct std::_List_const_iterator< _Tp >

A list::const_iterator.

All the functions are op overloads.

Definition at line 200 of file stl_list.h.

The documentation for this struct was generated from the following file:

- stl_list.h

4.539 std::_List_iterator< _Tp > Struct Template Reference

Public Types

- typedef _List_node< _Tp > **_Node**
- typedef _List_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef _Tp ∗ **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_iterator** (__detail::_List_node_base ∗__x)
- bool **operator!=** (const _Self &__x) const
- reference **operator∗** () const
- _Self & **operator++** ()
- _Self **operator++** (int)
- _Self & **operator--** ()
- _Self **operator--** (int)
- pointer **operator->** () const
- bool **operator==** (const _Self &__x) const

Public Attributes

- __detail::_List_node_base ∗ **_M_node**

4.539.1 Detailed Description

template<typename _Tp>struct std::_List_iterator< _Tp >

A list::iterator.

All the functions are op overloads.

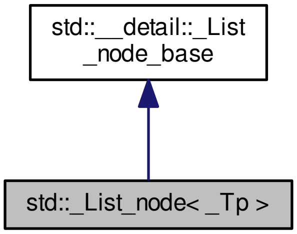
Definition at line 125 of file stl_list.h.

The documentation for this struct was generated from the following file:

- stl_list.h

4.540 std::_List_node< _Tp > Struct Template Reference

Inheritance diagram for std::_List_node< _Tp >:



Public Member Functions

- template<typename... _Args>
  _List_node (_Args &&... __args)
- void **_M_hook** (_List_node_base ∗const __position) noexcept
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base ∗const __first, _List_node_base ∗const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _Tp _M_data
- _List_node_base ∗ **_M_next**
- _List_node_base ∗ **_M_prev**

4.540.1 Detailed Description

template<typename _Tp>struct std::_List_node< _Tp >

An actual node in the list.

Definition at line 106 of file stl_list.h.

4.540.2 Member Data Documentation

4.540.2.1 template<typename _Tp> _Tp std::_List_node< _Tp >::_M_data

< User's data.

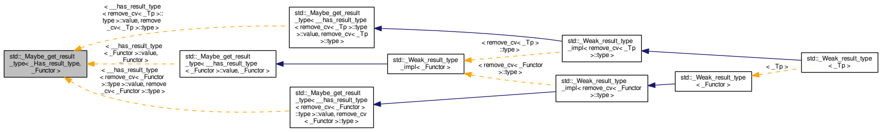
Definition at line 109 of file stl_list.h.

The documentation for this struct was generated from the following file:

- stl_list.h

4.541 std::_Maybe_get_result_type< _Has_result_type, _Functor > Struct Template Reference

Inheritance diagram for std::_Maybe_get_result_type< _Has_result_type, _Functor >:



4.541.1 Detailed Description

template<bool _Has_result_type, typename _Functor>struct std::_Maybe_get_result_type< _Has_result_type, _Functor >

If we have found a result_type, extract it.

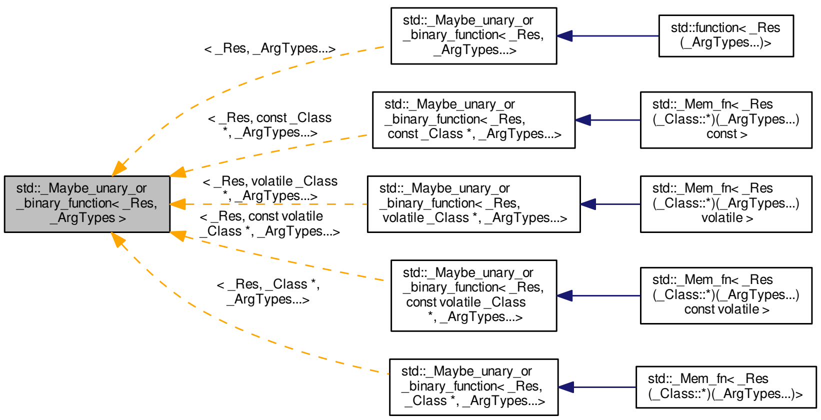
Definition at line 74 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.542 std::_Maybe_unary_or_binary_function< _Res, _ArgTypes > Struct Template Reference

Inheritance diagram for std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >:



4.542.1 Detailed Description

template<typename _Res, typename... _ArgTypes>struct std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >

Derives from `unary_function` or `binary_function`, or perhaps nothing, depending on the number of arguments provided. The primary template is the basis case, which derives nothing.

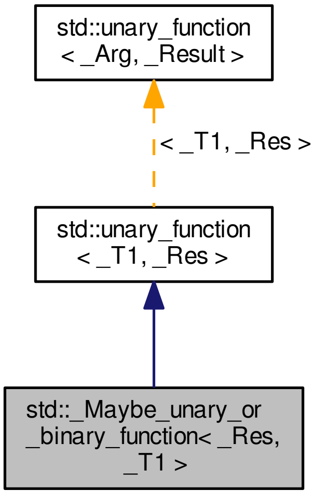
Definition at line 527 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.543 std::_Maybe_unary_or_binary_function< _Res, _T1 > Struct Template Reference

Inheritance diagram for std::_Maybe_unary_or_binary_function< _Res, _T1 >:



Public Types

- typedef _T1 argument_type
- typedef _Res result_type

4.543.1 Detailed Description

template<typename _Res, typename _T1>struct std::_Maybe_unary_or_binary_function< _Res, _T1 >

Derives from `unary_function`, as appropriate.

Definition at line 531 of file functional.

4.543.2 Member Typedef Documentation

4.543.2.1 typedef _T1 std::unary_function< _T1 , _Res >::argument_type `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.543.2.2 typedef _Res std::unary_function< _T1 , _Res >::result_type `[inherited]`

`result_type` is the return type

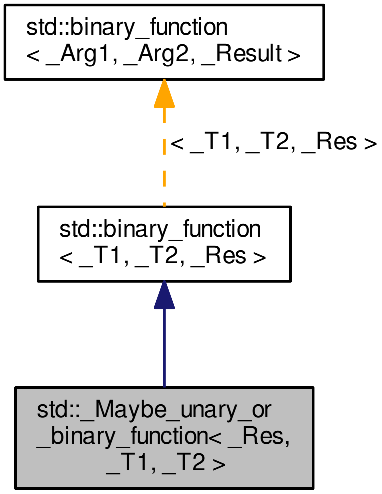
Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- functional

4.544 std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 > Struct Template Reference

Inheritance diagram for std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef _Res result_type
- typedef _T2 second_argument_type

4.544.1 Detailed Description

template<typename _Res, typename _T1, typename _T2>struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >

Derives from `binary_function`, as appropriate.

Definition at line 536 of file functional.

4.544.2 Member Typedef Documentation

4.544.2.1 typedef _T1 std::binary_function< _T1 , _T2 , _Res >::first_argument_type `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.544.2.2 typedef _Res std::binary_function< _T1 , _T2 , _Res >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.544.2.3 typedef _T2 std::binary_function< _T1 , _T2 , _Res >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- functional

4.545 std::_Maybe_wrap_member_pointer< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

Static Public Member Functions

- static const _Tp & **__do_wrap** (const _Tp &__x)
- static _Tp && **__do_wrap** (_Tp &&__x)

4.545.1 Detailed Description

template<typename _Tp>struct std::_Maybe_wrap_member_pointer< _Tp >

Maps member pointers into instances of _Mem_fn but leaves all other function objects untouched. Used by tr1::bind(). The primary template handles the non–member-pointer case.

Definition at line 1223 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.546 std::_Maybe_wrap_member_pointer< _Tp _Class::∗ > Struct Template Reference

Public Types

- typedef _Mem_fn< _Tp _Class::∗ > **type**

Static Public Member Functions

- static type **__do_wrap** (_Tp _Class::∗__pm)

4.546.1 Detailed Description

template<typename _Tp, typename _Class>struct std::_Maybe_wrap_member_pointer< _Tp _Class::∗ >

Maps member pointers into instances of _Mem_fn but leaves all other function objects untouched. Used by tr1::bind(). This partial specialization handles the member pointer case.

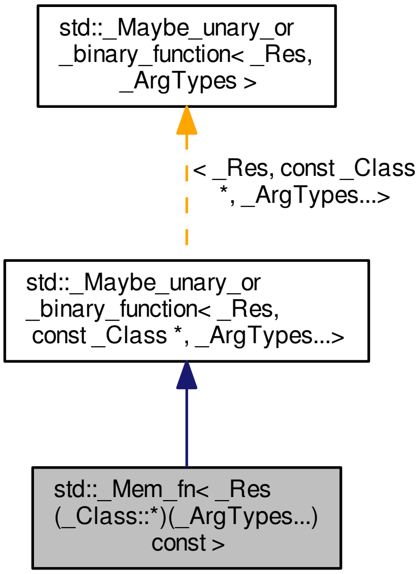
Definition at line 1242 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.547 std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const > Class Template Reference

Inheritance diagram for std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const >:



Public Types

- typedef _Res **result_type**

Public Member Functions

- **_Mem_fn** (_Functor __pmf)
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (const _Class &__object, _Args &&...__args) const

- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (const _Class &&__object, _Args &&...__args) const
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (const _Class ∗__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>
  _Res **operator()** (_Tp &&__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>
  _Res **operator()** (reference_wrapper< _Tp > __ref, _Args &&...__args) const

4.547.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const >

Implementation of `mem_fn` for const member function pointers.

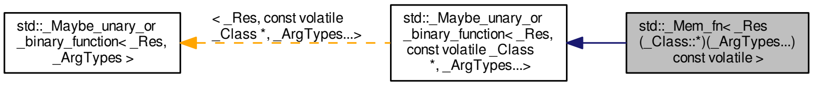
Definition at line 625 of file functional.

The documentation for this class was generated from the following file:

- functional

4.548 std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const volatile > Class Template Reference

Inheritance diagram for std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const volatile >:



Public Types

- typedef _Res **result_type**

Public Member Functions

- **_Mem_fn** (_Functor __pmf)
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (const volatile _Class &__object, _Args &&...__args) const
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (const volatile _Class &&__object, _Args &&...__args) const
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (const volatile _Class ∗__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>
  _Res **operator()** (_Tp &&__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>
  _Res **operator()** (reference_wrapper< _Tp > __ref, _Args &&...__args) const

4.548.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const volatile >

Implementation of `mem_fn` for const volatile member function pointers.

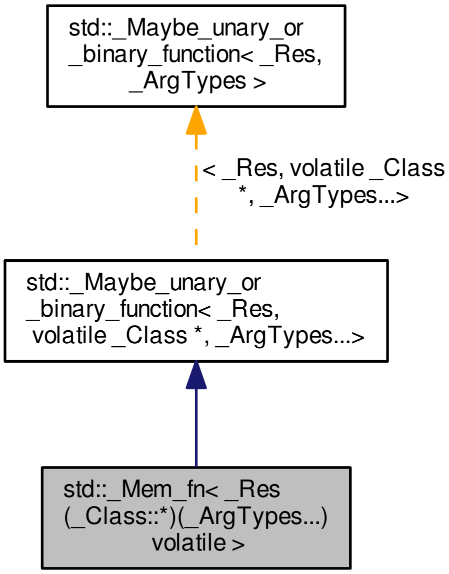
Definition at line 784 of file functional.

The documentation for this class was generated from the following file:

- functional

4.549 std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) volatile > Class Template Reference

Inheritance diagram for std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) volatile >:



Public Types

- typedef _Res **result_type**

Public Member Functions

- **_Mem_fn** (_Functor __pmf)
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (volatile _Class &__object, _Args &&...__args) const

- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (volatile _Class &&__object, _Args &&...__args) const
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (volatile _Class ∗__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>
  _Res **operator()** (_Tp &&__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>
  _Res **operator()** (reference_wrapper< _Tp > __ref, _Args &&...__args) const

4.549.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) volatile >

Implementation of `mem_fn` for volatile member function pointers.

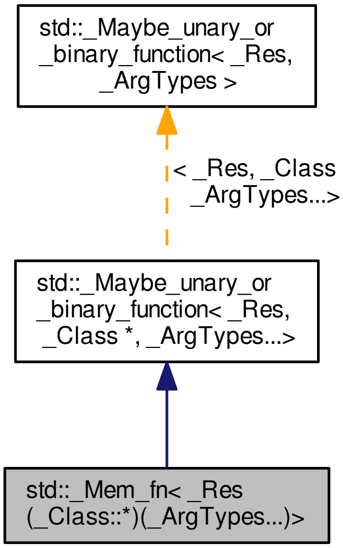
Definition at line 704 of file functional.

The documentation for this class was generated from the following file:

- functional

4.550 std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...)> Class Template Reference

Inheritance diagram for std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...)>:



Public Types

- typedef _Res **result_type**

Public Member Functions

- **_Mem_fn** (_Functor __pmf)
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (_Class &__object, _Args &&...__args) const
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (_Class &&__object, _Args &&...__args) const
- template<typename... _Args, typename _Req = _RequireValidArgs<_Args...>>
  _Res **operator()** (_Class ∗__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs2<_Tp, _Args...>>
  _Res **operator()** (_Tp &&__object, _Args &&...__args) const
- template<typename _Tp , typename... _Args, typename _Req = _RequireValidArgs3<_Tp, _Args...>>
  _Res **operator()** (reference_wrapper< _Tp > __ref, _Args &&...__args) const

4.550.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...)>

Implementation of `mem_fn` for member function pointers.

Definition at line 541 of file functional.

The documentation for this class was generated from the following file:

- functional

4.551 std::_Mu< _Arg, false, false > Class Template Reference

Public Member Functions

- template<typename _CVArg , typename _Tuple >
  _CVArg && **operator()** (_CVArg &&__arg, _Tuple &) const volatile

4.551.1 Detailed Description

template<typename _Arg>class std::_Mu< _Arg, false, false >

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are the same as the cv-qualifiers on the _Mu object. [TR1 3.6.3/5 bullet 4]

Definition at line 1199 of file functional.

The documentation for this class was generated from the following file:

- functional

4.552 std::_Mu< _Arg, false, true > Class Template Reference

Public Member Functions

- template<typename _Tuple >
  result< _Mu(_Arg, _Tuple)>::type **operator()** (const volatile _Arg &, _Tuple &__tuple) const volatile

4.552.1 Detailed Description

template<typename _Arg>class std::_Mu< _Arg, false, true >

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. [TR1 3.6.3/5 bullet 3]

Definition at line 1165 of file functional.

The documentation for this class was generated from the following file:

- functional

4.553 std::_Mu< _Arg, true, false > Class Template Reference

Public Member Functions

- template<typename _CVArg , typename... _Args>
  auto **operator()** (_CVArg &__arg, tuple< _Args...> &__tuple) const volatile-> decltype(__arg(declval< _Args >()...))

4.553.1 Detailed Description

template<typename _Arg>class std::_Mu< _Arg, true, false >

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). [TR1 3.6.3/5 bullet 2]

Definition at line 1131 of file functional.

The documentation for this class was generated from the following file:

- functional

4.554 std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference

Public Types

- typedef _Tp & **result_type**

Public Member Functions

- template<typename _CVRef , typename _Tuple >
  result_type **operator()** (_CVRef &__arg, _Tuple &) const volatile

4.554.1 Detailed Description

template<typename _Tp>class std::_Mu< reference_wrapper< _Tp >, false, false >

If the argument is reference_wrapper<_Tp>, returns the underlying reference. [TR1 3.6.3/5 bullet 1]

Definition at line 1110 of file functional.

The documentation for this class was generated from the following file:

- functional

4.555 std::_Placeholder< _Num > Struct Template Reference

4.555.1 Detailed Description

template<int _Num>struct std::_Placeholder< _Num >

The type of placeholder objects defined by libstdc++.

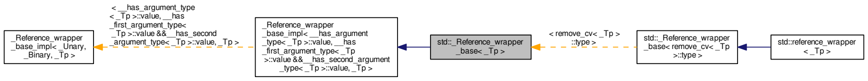
Definition at line 989 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.556 std::_Reference_wrapper_base< _Tp > Struct Template Reference

Inheritance diagram for std::_Reference_wrapper_base< _Tp >:



4.556.1 Detailed Description

template<typename _Tp>struct std::_Reference_wrapper_base< _Tp >

Derives from unary_function or binary_function when it can. Specializations handle all of the easy cases. The primary template determines what to do with a class type, which may derive from both unary_function and binary_function.

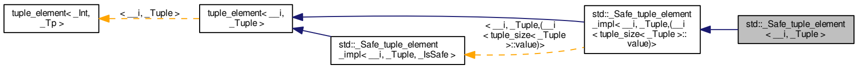
Definition at line 315 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.557 std::_Safe_tuple_element< __i, _Tuple > Struct Template Reference

Inheritance diagram for std::_Safe_tuple_element< __i, _Tuple >:



4.557.1 Detailed Description

template<std::size_t __i, typename _Tuple>struct std::_Safe_tuple_element< __i, _Tuple >

Like tuple_element, but returns `_No_tuple_element` when tuple_element would return an error.

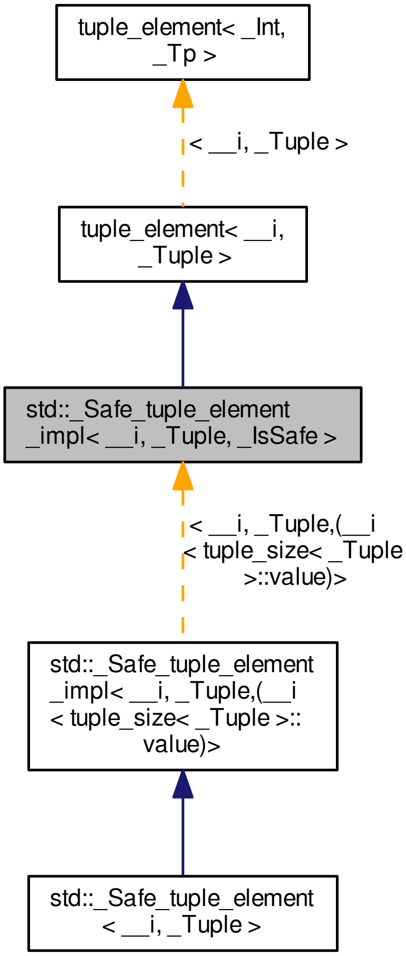
Definition at line 1084 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.558 std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe > Struct Template Reference

Inheritance diagram for std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >:



4.558.1 Detailed Description

template<std::size_t __i, typename _Tuple, bool _IsSafe>struct std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >

Implementation helper for _Safe_tuple_element. This primary template handles the case where it is safe to use `tuple_element`.

Definition at line 1065 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.559 std::_Safe_tuple_element_impl< __i, _Tuple, false > Struct Template Reference

Public Types

- typedef _No_tuple_element **type**

4.559.1 Detailed Description

template<std::size_t __i, typename _Tuple>struct std::_Safe_tuple_element_impl< __i, _Tuple, false >

Implementation helper for _Safe_tuple_element. This partial specialization handles the case where it is not safe to use `tuple_element`. We just return `_No_tuple_element`.

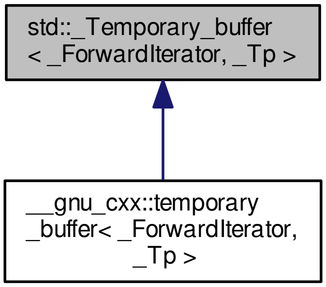
Definition at line 1074 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.560 std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference

Inheritance diagram for std::_Temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type ∗ **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- _Temporary_buffer (_ForwardIterator __first, _ForwardIterator __last)
- iterator begin ()
- iterator end ()
- size_type requested_size () const
- size_type size () const

Protected Attributes

- pointer **_M_buffer**
- size_type **_M_len**
- size_type **_M_original_len**

4.560.1 Detailed Description

template<typename _ForwardIterator, typename _Tp>class std::_Temporary_buffer< _ForwardIterator, _Tp >

This class is used in two places: stl_algo.h and ext/memory, where it is wrapped as the temporary_buffer class. See temporary_buffer docs for more notes.

Definition at line 122 of file stl_tempbuf.h.

4.560.2 Constructor & Destructor Documentation

4.560.2.1 template<typename _ForwardIterator , typename _Tp > std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (_ForwardIterator *__first,* _ForwardIterator *__last*)

Constructs a temporary buffer of a size somewhere between zero and the size of the given range.

Definition at line 244 of file stl_tempbuf.h.

References std::pair< _T1, _T2 >::first, std::get_temporary_buffer(), std::return_temporary_buffer(), and std::pair< _T1, _T2 >::second.

4.560.3 Member Function Documentation

4.560.3.1 template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ( ) [inline]

As per Table mumble.

Definition at line 151 of file stl_tempbuf.h.

Referenced by std::inplace_merge(), std::stable_partition(), and std::stable_sort().

4.560.3.2 template<typename _ForwardIterator, typename _Tp> iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ( ) [inline]

As per Table mumble.

Definition at line 156 of file stl_tempbuf.h.

4.560.3.3 template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size ( ) const [inline]

Returns the size requested by the constructor; may be >size().

Definition at line 146 of file stl_tempbuf.h.

Referenced by std::stable_partition().

4.560.3.4 template<typename _ForwardIterator, typename _Tp> size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size ( ) const [inline]

As per Table mumble.

Definition at line 141 of file stl_tempbuf.h.

Referenced by std::inplace_merge(), std::stable_partition(), and std::stable_sort().

The documentation for this class was generated from the following file:

- stl_tempbuf.h

4.561 std::_Tuple_impl< _Idx > Struct Template Reference

Public Member Functions

- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t, const _Alloc &)
- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t, const _Alloc &, const _Tuple_impl &)
- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t, const _Alloc &, _Tuple_impl &&)

Protected Member Functions

- void **_M_swap** (_Tuple_impl &) noexcept

Friends

- template<std::size_t , typename... >
  class **_Tuple_impl**

4.561.1 Detailed Description

template<std::size_t _Idx>struct std::_Tuple_impl< _Idx >

Zero-element tuple implementation. This is the basis case for the inheritance recursion.

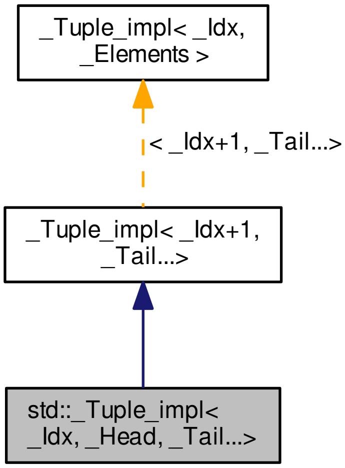
Definition at line 191 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.562 std::_Tuple_impl< _Idx, _Head, _Tail...> Struct Template Reference

Inheritance diagram for std::_Tuple_impl< _Idx, _Head, _Tail...>:



Public Types

- typedef _Head_base< _Idx, _Head, __empty_not_final < _Head >::value > **_Base**
- typedef _Tuple_impl< _Idx+1, _Tail...> **_Inherited**

Public Member Functions

- constexpr **_Tuple_impl** (const _Head &__head, const _Tail &...__tail)
- template<typename _UHead , typename... _UTail, typename = typename enable_if<sizeof...(_Tail) == sizeof...(_UTail)>::type>
  constexpr **_Tuple_impl** (_UHead &&__head, _UTail &&...__tail)
- constexpr **_Tuple_impl** (const _Tuple_impl &)=default
- constexpr **_Tuple_impl** (_Tuple_impl &&__in) noexcept(__and_< is_nothrow_move_constructible< _Head >
- template<typename... _UElements>
  constexpr **_Tuple_impl** (const _Tuple_impl< _Idx, _UElements...> &__in)
- template<typename _UHead , typename... _UTails>
  constexpr **_Tuple_impl** (_Tuple_impl< _Idx, _UHead, _UTails...> &&__in)

- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, const _Head &__head, const _Tail &... __tail)
- template<typename _Alloc , typename _UHead , typename... _UTail, typename = typename enable_if<sizeof...(_Tail) == sizeof...(_UTail)>::type>
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, _UHead &&__head, _UTail &&... __tail)
- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, const _Tuple_impl &__in)
- template<typename _Alloc >
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, _Tuple_impl &&__in)
- template<typename _Alloc , typename... _UElements>
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, const _Tuple_impl< _Idx, _UElements...> &__in)
- template<typename _Alloc , typename _UHead , typename... _UTails>
  _Tuple_impl (allocator_arg_t __tag, const _Alloc &__a, _Tuple_impl< _Idx, _UHead, _UTails...> &&__in)
- constexpr
  is_nothrow_move_constructible
  < _Inherited > **_Base** (std::forward< _Head >(_M_head(__in)))
- _Tuple_impl & **operator=** (const _Tuple_impl &__in)
- _Tuple_impl & **operator=** (_Tuple_impl &&__in) noexcept(__and_< is_nothrow_move_assignable< _Head >
- template<typename... _UElements>
  _Tuple_impl & **operator=** (const _Tuple_impl< _Idx, _UElements...> &__in)
- template<typename _UHead , typename... _UTails>
  _Tuple_impl & **operator=** (_Tuple_impl< _Idx, _UHead, _UTails...> &&__in)

Static Public Member Functions

- static constexpr _Head & **_M_head** (_Tuple_impl &__t) noexcept
- static constexpr const _Head & **_M_head** (const _Tuple_impl &__t) noexcept
- static constexpr _Inherited & **_M_tail** (_Tuple_impl &__t) noexcept
- static constexpr const _Inherited & **_M_tail** (const _Tuple_impl &__t) noexcept

Public Attributes

- _M_tail **this**
- return ∗ **this**

Protected Member Functions

- void **_M_swap** (_Tuple_impl &__in) noexcept(noexcept(swap(std

Friends

- template<std::size_t , typename... >
  class **_Tuple_impl**

4.562.1 Detailed Description

template<std::size_t _Idx, typename _Head, typename... _Tail>struct std::_Tuple_impl< _Idx, _Head, _Tail...>

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

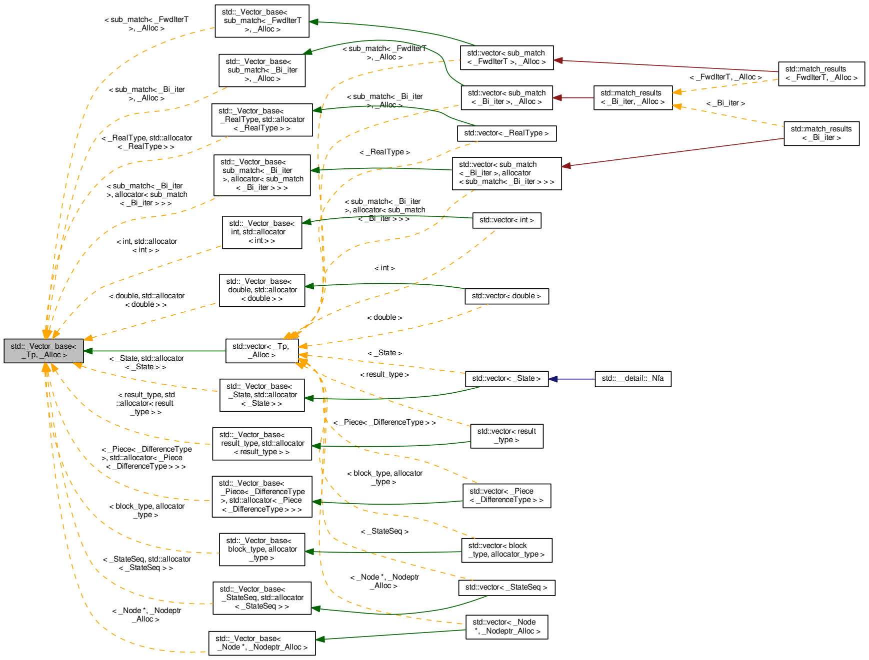
Definition at line 229 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.563 std::_Vector_base< _Tp, _Alloc > Struct Template Reference

Inheritance diagram for std::_Vector_base< _Tp, _Alloc >:



Public Types

- typedef __gnu_cxx::__alloc_traits < _Alloc >::template rebind < _Tp >::other **_Tp_alloc_type**
- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__alloc_traits < _Tp_alloc_type >::pointer **pointer**

Public Member Functions

- **_Vector_base** (const allocator_type &__a)
- **_Vector_base** (size_t __n)
- **_Vector_base** (size_t __n, const allocator_type &__a)
- **_Vector_base** (_Tp_alloc_type &&__a)
- **_Vector_base** (_Vector_base &&__x)
- **_Vector_base** (_Vector_base &&__x, const allocator_type &__a)
- pointer **_M_allocate** (size_t __n)
- void **_M_deallocate** (pointer __p, size_t __n)
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- allocator_type **get_allocator** () const noexcept

Public Attributes

- _Vector_impl **_M_impl**

4.563.1 Detailed Description

template<typename _Tp, typename _Alloc>struct std::_Vector_base< _Tp, _Alloc >

See bits/stl_deque.h's _Deque_base for an explanation.

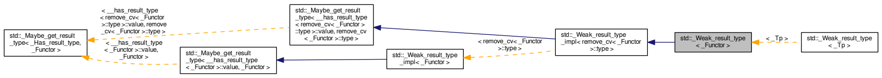
Definition at line 72 of file stl_vector.h.

The documentation for this struct was generated from the following file:

- stl_vector.h

4.564 std::_Weak_result_type< _Functor > Struct Template Reference

Inheritance diagram for std::_Weak_result_type< _Functor >:



4.564.1 Detailed Description

template<typename _Functor>struct std::_Weak_result_type< _Functor >

Strip top-level cv-qualifiers from the function object and let _Weak_result_type_impl perform the real work.

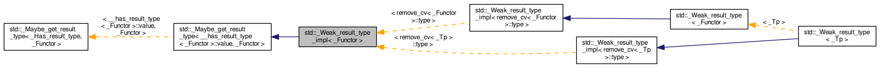
Definition at line 184 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.565 std::_Weak_result_type_impl< _Functor > Struct Template Reference

Inheritance diagram for std::_Weak_result_type_impl< _Functor >:



4.565.1 Detailed Description

template<typename _Functor>struct std::_Weak_result_type_impl< _Functor >

Base class for any function object that has a weak result type, as defined in 3.3/3 of TR1.

Definition at line 86 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.566 std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)> Struct Template Reference

Public Types

- typedef _Res **result_type**

4.566.1 Detailed Description

template<typename _Res, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)>

Retrieve the result type for a function reference.

Definition at line 125 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.567 std::_Weak_result_type_impl< _Res(∗)(_ArgTypes...)> Struct Template Reference

Public Types

- typedef _Res **result_type**

4.567.1 Detailed Description

template<typename _Res, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(∗)(_ArgTypes...)>

Retrieve the result type for a function pointer.

Definition at line 134 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.568 std::_Weak_result_type_impl< _Res(_ArgTypes...)> Struct Template Reference

Public Types

- typedef _Res **result_type**

4.568.1 Detailed Description

template<typename _Res, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_ArgTypes...)>

Retrieve the result type for a function type.

Definition at line 92 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.569 std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const > Struct Template Reference

Public Types

- typedef _Res **result_type**

4.569.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const >

Retrieve result type for a const member function pointer.

Definition at line 152 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.570 std::_Weak_result_type_impl< _Res(_Class::*)(_ArgTypes...) const volatile > Struct Template Reference

Public Types

- typedef _Res **result_type**

4.570.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) const volatile >

Retrieve result type for a const volatile member function pointer.

Definition at line 170 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.571 std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) volatile > Struct Template Reference

Public Types

- typedef _Res **result_type**

4.571.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) volatile >

Retrieve result type for a volatile member function pointer.

Definition at line 161 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.572 std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...)> Struct Template Reference

Public Types

- typedef _Res **result_type**

4.572.1 Detailed Description

template<typename _Res, typename _Class, typename... _ArgTypes>struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...)>

Retrieve result type for a member function pointer.

Definition at line 143 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.573 std::add_const< _Tp > Struct Template Reference

Public Types

- typedef _Tp const **type**

4.573.1 Detailed Description

template<typename _Tp>struct std::add_const< _Tp >

add_const

Definition at line 1353 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.574 std::add_cv< _Tp > Struct Template Reference

Public Types

- typedef add_const< typename add_volatile< _Tp >::type > ::type **type**

4.574.1 Detailed Description

template<typename _Tp>struct std::add_cv< _Tp >

add_cv

Definition at line 1363 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.575 std::add_lvalue_reference< _Tp > Struct Template Reference

Inherits std::__add_lvalue_reference_helper< _Tp, bool, bool >.

Public Types

- typedef _Tp **type**

4.575.1 Detailed Description

template<typename _Tp>struct std::add_lvalue_reference< _Tp >

add_lvalue_reference

Definition at line 1402 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.576 std::add_pointer< _Tp > Struct Template Reference

Public Types

- typedef remove_reference< _Tp >
  ::type ∗ **type**

4.576.1 Detailed Description

template<typename _Tp>struct std::add_pointer< _Tp >

add_pointer

Definition at line 1667 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.577 std::add_rvalue_reference< _Tp > Struct Template Reference

Inherits std::__add_rvalue_reference_helper< _Tp, bool >.

Public Types

- typedef _Tp **type**

4.577.1 Detailed Description

template<typename _Tp>struct std::add_rvalue_reference< _Tp >

add_rvalue_reference

Definition at line 1418 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.578 std::add_volatile< _Tp > Struct Template Reference

Public Types

- typedef _Tp volatile **type**

4.578.1 Detailed Description

template<typename _Tp>struct std::add_volatile< _Tp >

add_volatile

Definition at line 1358 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.579 std::adopt_lock_t Struct Reference

4.579.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.

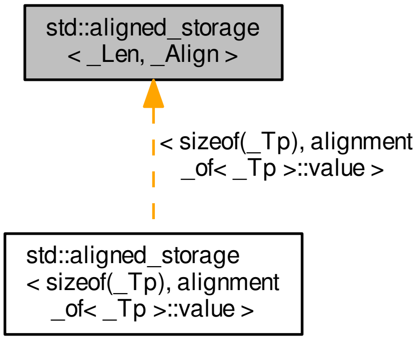
Definition at line 396 of file mutex.

The documentation for this struct was generated from the following file:

- mutex

4.580 std::aligned_storage< _Len, _Align > Struct Template Reference

Inheritance diagram for std::aligned_storage< _Len, _Align >:



4.580.1 Detailed Description

template<std::size_t _Len, std::size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>struct std::aligned_storage< _Len, _Align >

Alignment type.

The value of _Align is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than _Len (3.9). The member typedef type shall be a POD type suitable for use as uninitialized storage for any object whose size is at most _Len and whose alignment is a divisor of _Align.

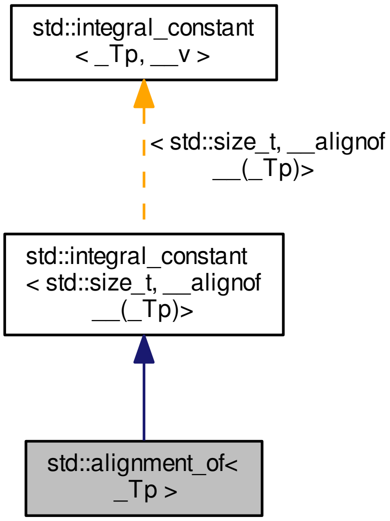
Definition at line 1693 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.581 std::alignment_of< _Tp > Struct Template Reference

Inheritance diagram for std::alignment_of< _Tp >:



Public Types

- typedef integral_constant
  < std::size_t, __v > **type**
- typedef std::size_t **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr std::size_t **value**

4.581.1 Detailed Description

template<typename _Tp>struct std::alignment_of< _Tp >

alignment_of

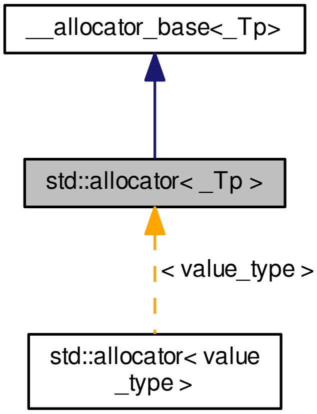
Definition at line 1238 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.582 std::allocator< _Tp > Struct Template Reference

Inheritance diagram for std::allocator< _Tp >:



Public Types

- typedef const _Tp ∗ **const_pointer**
- typedef const _Tp & **const_reference**
- typedef ptrdiff_t **difference_type**
- typedef _Tp ∗ **pointer**
- typedef true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **allocator** (const allocator &__a) throw ()
- template<typename _Tp1 >
  allocator (const allocator< _Tp1 > &) throw ()
- pointer **address** (reference __x) const noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **allocate** (size_type __n, const void ∗=0)

- template<typename _Up , typename... _Args>
  void **construct** (_Up ∗__p, _Args &&...__args)
- void **deallocate** (pointer __p, size_type)
- template<typename _Up >
  void **destroy** (_Up ∗__p)
- size_type **max_size** () const noexcept

4.582.1 Detailed Description

template<typename _Tp>struct std::allocator< _Tp >

The *standard* allocator, as per [20.4].

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt04ch11.html` for further details.

Template Parameters

_Tp	Type of allocated object.

Destroy a range of objects using the supplied allocator. For nondefault allocators we do not optimize away invocation of destroy() even if _Tp has a trivial destructor.

Definition at line 99 of file allocator.h.

The documentation for this struct was generated from the following file:

- allocator.h

4.583 std::allocator< void > Class Template Reference

Public Types

- typedef const void ∗ **const_pointer**
- typedef ptrdiff_t **difference_type**
- typedef void ∗ **pointer**
- typedef true_type **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef void **value_type**

4.583.1 Detailed Description

template<>class std::allocator< void >

allocator<void> specialization.

Definition at line 70 of file allocator.h.

The documentation for this class was generated from the following file:

- allocator.h

4.584 std::allocator_arg_t Struct Reference

4.584.1 Detailed Description

[allocator.tag]

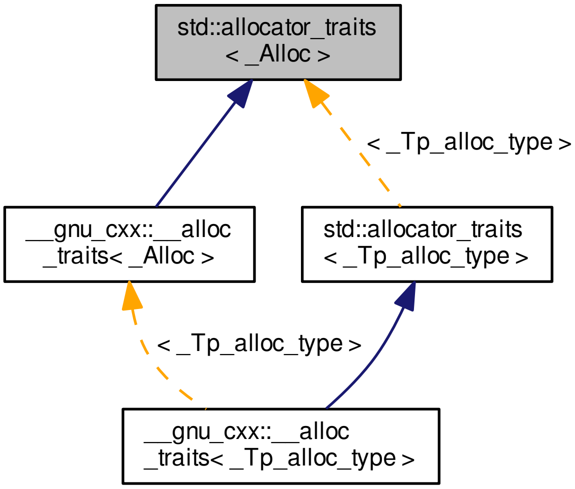
Definition at line 39 of file uses_allocator.h.

The documentation for this struct was generated from the following file:

- uses_allocator.h

4.585 std::allocator_traits< _Alloc > Struct Template Reference

Inheritance diagram for std::allocator_traits< _Alloc >:



Public Types

- typedef _Alloc allocator_type
- typedef __const_pointer const_pointer
- typedef __const_void_pointer const_void_pointer
- typedef __difference_type difference_type
- typedef __pointer pointer
- typedef __propagate_on_container_copy_assignment propagate_on_container_copy_assignment
- typedef __propagate_on_container_move_assignment propagate_on_container_move_assignment
- typedef __propagate_on_container_swap propagate_on_container_swap

- template<typename _Tp >
  using **rebind_alloc** = typename __alloctr_rebind< _Alloc, _Tp >::__type
- template<typename _Tp >
  using **rebind_traits** = allocator_traits< rebind_alloc< _Tp >>
- typedef __size_type size_type
- typedef _Alloc::value_type value_type
- typedef __void_pointer void_pointer

Static Public Member Functions

- static pointer allocate (_Alloc &__a, size_type __n)
- static pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)
- template<typename _Tp , typename... _Args>
  static auto construct (_Alloc &__a, _Tp ∗__p, _Args &&...__args) -> decltype(_S_construct(__a, __p, std::forward< _Args >(__args)...))
- static void deallocate (_Alloc &__a, pointer __p, size_type __n)
- template<class _Tp >
  static void destroy (_Alloc &__a, _Tp ∗__p)
- static size_type max_size (const _Alloc &__a)
- static _Alloc select_on_container_copy_construction (const _Alloc &__rhs)

4.585.1 Detailed Description

template<typename _Alloc>struct std::allocator_traits< _Alloc >

Uniform interface to all allocator types.

Definition at line 87 of file bits/alloc_traits.h.

4.585.2 Member Typedef Documentation

4.585.2.1 template<typename _Alloc> typedef _Alloc std::allocator_traits< _Alloc >::allocator_type

The allocator type.

Definition at line 90 of file bits/alloc_traits.h.

4.585.2.2 template<typename _Alloc> typedef __const_pointer std::allocator_traits< _Alloc >::const_pointer

The allocator's const pointer type.

`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

Definition at line 120 of file bits/alloc_traits.h.

4.585.2.3 template<typename _Alloc> typedef __const_void_pointer std::allocator_traits< _Alloc >::const_void_pointer

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 142 of file bits/alloc_traits.h.

4.585.2.4 template<typename _Alloc> typedef __difference_type std::allocator_traits< _Alloc >::difference_type

The allocator's difference type.

`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

Definition at line 153 of file bits/alloc_traits.h.

4.585.2.5 template<typename _Alloc> typedef __pointer std::allocator_traits< _Alloc >::pointer

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

Definition at line 109 of file bits/alloc_traits.h.

4.585.2.6 template<typename _Alloc> typedef __propagate_on_container_copy_assignment std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 176 of file bits/alloc_traits.h.

4.585.2.7 template<typename _Alloc> typedef __propagate_on_container_move_assignment std::allocator_traits< _Alloc >::propagate_on_container_move_assignment

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 188 of file bits/alloc_traits.h.

4.585.2.8 template<typename _Alloc> typedef __propagate_on_container_swap std::allocator_traits< _Alloc >::propagate_on_container_swap

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 199 of file bits/alloc_traits.h.

4.585.2.9 template<typename _Alloc> typedef __size_type std::allocator_traits< _Alloc >::size_type

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

Definition at line 164 of file bits/alloc_traits.h.

4.585.2.10 template<typename _Alloc> typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type

The allocated type.

Definition at line 92 of file bits/alloc_traits.h.

4.585.2.11 template<typename _Alloc> typedef __void_pointer std::allocator_traits< _Alloc >::void_pointer

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 131 of file bits/alloc_traits.h.

4.585.3 Member Function Documentation

4.585.3.1 template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate (_Alloc & __a, size_type __n) [inline], [static]

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.

Calls `a.allocate(n)`

Definition at line 352 of file bits/alloc_traits.h.

4.585.3.2 template<typename _Alloc> static pointer std::allocator_traits< _Alloc >::allocate (_Alloc & __a, size_type __n, const_void_pointer __hint) [inline], [static]

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 367 of file bits/alloc_traits.h.

4.585.3.3 template<typename _Alloc> template<typename _Tp , typename... _Args> static auto std::allocator_traits< _Alloc >::construct (_Alloc & __a, _Tp ∗ __p, _Args &&... __args) -> decltype(_S_construct(__a, __p, std::forward< _Args>(__args)...)) [inline], [static]

Construct an object of type *_Tp*.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

Definition at line 393 of file bits/alloc_traits.h.

4.585.3.4 template<typename _Alloc> static void **std::allocator_traits**< _Alloc >::deallocate (_Alloc & *__a*, **pointer** *__p*, **size_type** *__n*) [inline], [static]

Deallocate memory.

Parameters

__a	An allocator.
__p	Pointer to the memory to deallocate.
__n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

Definition at line 378 of file bits/alloc_traits.h.

4.585.3.5 template<typename _Alloc> template<class _Tp > static void **std::allocator_traits**< _Alloc >::destroy (_Alloc & *__a*, _Tp * *__p*) [inline], [static]

Destroy an object of type *_Tp*.

Parameters

__a	An allocator.
__p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 406 of file bits/alloc_traits.h.

4.585.3.6 template<typename _Alloc> static **size_type std::allocator_traits**< _Alloc >::max_size (const _Alloc & *__a*) [inline], [static]

The maximum supported allocation size.

Parameters

__a	An allocator.

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 417 of file bits/alloc_traits.h.

Referenced by std::forward_list< _Tp, _Alloc >::max_size().

4.585.3.7 template<typename _Alloc> static _Alloc **std::allocator_traits**< _Alloc >::select_on_container_copy_construction (const _Alloc & *__rhs*) [inline], [static]

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

Definition at line 429 of file bits/alloc_traits.h.

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

4.586 std::array< _Tp, _Nm > Struct Template Reference

Public Types

- typedef ::__array_traits< _Tp, _Nm > **_AT_Type**
- typedef const value_type ∗ **const_iterator**
- typedef const value_type ∗ **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type ∗ **iterator**
- typedef value_type ∗ **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- reference **at** (size_type __n)
- constexpr const_reference **at** (size_type __n) const
- reference **back** ()
- constexpr const_reference **back** () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- pointer **data** () noexcept
- const_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- void **fill** (const value_type &__u)
- reference **front** ()

- constexpr const_reference **front** () const
- constexpr size_type **max_size** () const noexcept
- reference **operator[ ]** (size_type __n)
- constexpr const_reference **operator[ ]** (size_type __n) const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- constexpr size_type **size** () const noexcept
- void **swap** (array &__other) noexcept(noexcept(swap(std

Public Attributes

- _AT_Type::_Type **_M_elems**

4.586.1 Detailed Description

template<typename _Tp, std::size_t _Nm>struct std::array< _Tp, _Nm >

A standard container for storing a fixed size sequence of elements.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`.

Sets support random access iterators.

Template Parameters

Tp	Type of element. Required to be a complete type.
N	Number of elements.

Definition at line 81 of file array.

The documentation for this struct was generated from the following file:

- array

4.587 std::atomic< _Tp > Struct Template Reference

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (_Tp __i) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatilenoexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatilenoexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

- _Tp **exchange** (_Tp __i, memory_order _m=memory_order_seq_cst) noexcept
- _Tp **exchange** (_Tp __i, memory_order _m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- _Tp **load** (memory_order _m=memory_order_seq_cst) const noexcept
- _Tp **load** (memory_order _m=memory_order_seq_cst) const volatilenoexcept
- **operator _Tp** () const noexcept
- **operator _Tp** () const volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- _Tp **operator=** (_Tp __i) noexcept
- _Tp **operator=** (_Tp __i) volatilenoexcept
- void **store** (_Tp __i, memory_order _m=memory_order_seq_cst) noexcept
- void **store** (_Tp __i, memory_order _m=memory_order_seq_cst) volatilenoexcept

4.587.1 Detailed Description

template<typename _Tp>struct std::atomic< _Tp >

Generic atomic type, primary class template.

Template Parameters

_Tp	Type to be made atomic, must be trivally copyable.

Definition at line 161 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.588 std::atomic< _Tp ∗ > Struct Template Reference

Public Types

- typedef __atomic_base< _Tp ∗ > **__base_type**
- typedef _Tp ∗ **__pointer_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__pointer_type __p) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatilenoexcept

- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __pointer_type** () const noexcept
- **operator __pointer_type** () const volatilenoexcept
- __pointer_type **operator++** (int) noexcept
- __pointer_type **operator++** (int) volatilenoexcept
- __pointer_type **operator++** () noexcept
- __pointer_type **operator++** () volatilenoexcept
- __pointer_type **operator+=** (ptrdiff_t __d) noexcept
- __pointer_type **operator+=** (ptrdiff_t __d) volatilenoexcept
- __pointer_type **operator--** (int) noexcept
- __pointer_type **operator--** (int) volatilenoexcept
- __pointer_type **operator--** () noexcept
- __pointer_type **operator--** () volatilenoexcept
- __pointer_type **operator-=** (ptrdiff_t __d) noexcept
- __pointer_type **operator-=** (ptrdiff_t __d) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __pointer_type **operator=** (__pointer_type __p) noexcept
- __pointer_type **operator=** (__pointer_type __p) volatilenoexcept
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatilenoexcept

Public Attributes

- __base_type **_M_b**

4.588.1 Detailed Description

template<typename _Tp>struct std::atomic< _Tp ∗ >

Partial specialization for pointer types.

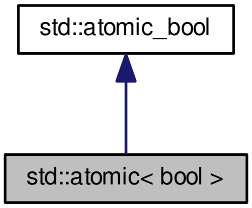
Definition at line 290 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.589 std::atomic< bool > Struct Template Reference

Inheritance diagram for std::atomic< bool >:



Public Types

- typedef atomic_bool **__base_type**
- typedef bool **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) volatile-noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) volatile-noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) volatile-noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) volatile-noexcept
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- bool **load** (memory_order __m=memory_order_seq_cst) const noexcept
- bool **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator bool** () const noexcept
- **operator bool** () const volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- void **store** (bool __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (bool __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.589.1 Detailed Description

template<>struct std::atomic< bool >

Explicit specialization for bool.

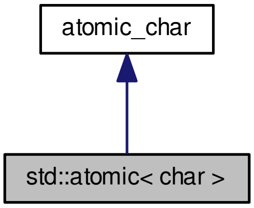
Definition at line 480 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.590 std::atomic< char > Struct Template Reference

Inheritance diagram for std::atomic< char >:



Public Types

- typedef atomic_char **__base_type**
- typedef char **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.590.1 Detailed Description

template<>struct std::atomic< char >

Explicit specialization for char.

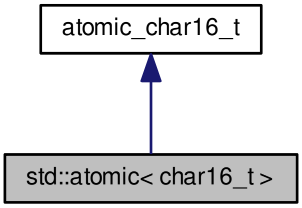
Definition at line 499 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.591 std::atomic< char16_t > Struct Template Reference

Inheritance diagram for std::atomic< char16_t >:



Public Types

- typedef atomic_char16_t **__base_type**
- typedef char16_t **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.591.1 Detailed Description

template<>struct std::atomic< char16_t >

Explicit specialization for char16_t.

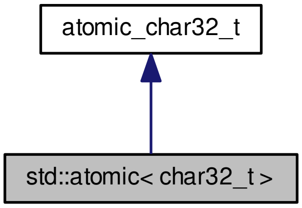
Definition at line 727 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.592 std::atomic< char32_t > Struct Template Reference

Inheritance diagram for std::atomic< char32_t >:



Public Types

- typedef atomic_char32_t **__base_type**
- typedef char32_t **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.592.1 Detailed Description

template<>struct std::atomic< char32_t >

Explicit specialization for char32_t.

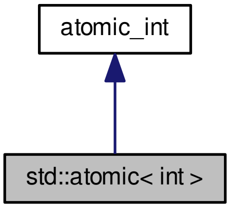
Definition at line 746 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.593 std::atomic< int > Struct Template Reference

Inheritance diagram for std::atomic< int >:



Public Types

- typedef atomic_int **__base_type**
- typedef int **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.593.1 Detailed Description

template<>struct std::atomic< int >

Explicit specialization for int.

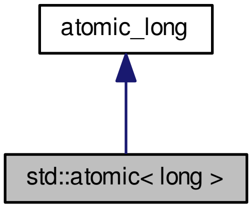
Definition at line 594 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.594 std::atomic< long > Struct Template Reference

Inheritance diagram for std::atomic< long >:



Public Types

- typedef atomic_long **__base_type**
- typedef long **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.594.1 Detailed Description

template<>struct std::atomic< long >

Explicit specialization for long.

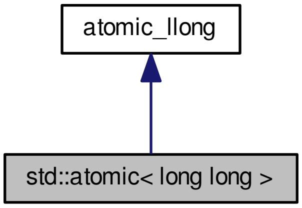
Definition at line 632 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.595 std::atomic< long long > Struct Template Reference

Inheritance diagram for std::atomic< long long >:



Public Types

- typedef atomic_llong **__base_type**
- typedef long long **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.595.1 Detailed Description

template<>struct std::atomic< long long >

Explicit specialization for long long.

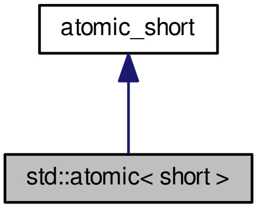
Definition at line 670 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.596 std::atomic< short > Struct Template Reference

Inheritance diagram for std::atomic< short >:



Public Types

- typedef atomic_short **__base_type**
- typedef short **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.596.1 Detailed Description

template<>struct std::atomic< short >

Explicit specialization for short.

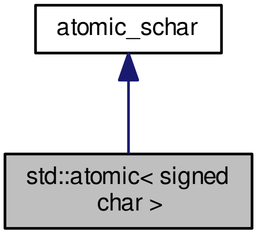
Definition at line 556 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.597 std::atomic< signed char > Struct Template Reference

Inheritance diagram for std::atomic< signed char >:



Public Types

- typedef atomic_schar **__base_type**
- typedef signed char **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.597.1 Detailed Description

template<>struct std::atomic< signed char >

Explicit specialization for signed char.

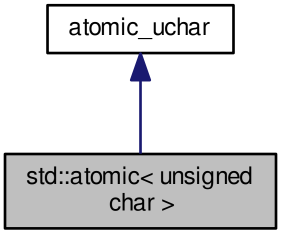
Definition at line 518 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.598 std::atomic< unsigned char > Struct Template Reference

Inheritance diagram for std::atomic< unsigned char >:



Public Types

- typedef atomic_uchar **__base_type**
- typedef unsigned char **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.598.1 Detailed Description

template<>struct std::atomic< unsigned char >

Explicit specialization for unsigned char.

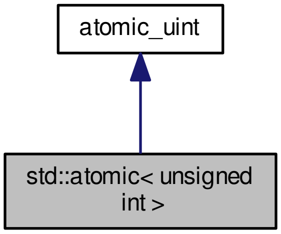
Definition at line 537 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.599 std::atomic< unsigned int > Struct Template Reference

Inheritance diagram for std::atomic< unsigned int >:



Public Types

- typedef atomic_uint **__base_type**
- typedef unsigned int **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.599.1 Detailed Description

template<>struct std::atomic< unsigned int >

Explicit specialization for unsigned int.

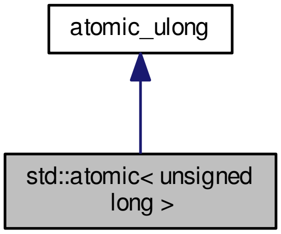
Definition at line 613 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.600 std::atomic< unsigned long > Struct Template Reference

Inheritance diagram for std::atomic< unsigned long >:



Public Types

- typedef atomic_ulong **__base_type**
- typedef unsigned long **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.600.1 Detailed Description

template<>struct std::atomic< unsigned long >

Explicit specialization for unsigned long.

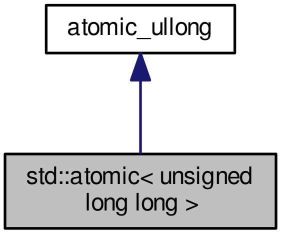
Definition at line 651 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.601 std::atomic< unsigned long long > Struct Template Reference

Inheritance diagram for std::atomic< unsigned long long >:



Public Types

- typedef atomic_ullong **__base_type**
- typedef unsigned long long **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.601.1 Detailed Description

template<>struct std::atomic< unsigned long long >

Explicit specialization for unsigned long long.

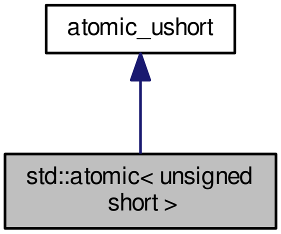
Definition at line 689 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.602 std::atomic< unsigned short > Struct Template Reference

Inheritance diagram for std::atomic< unsigned short >:



Public Types

- typedef atomic_ushort **__base_type**
- typedef unsigned short **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator\|=** (__int_type __i) noexcept
- __int_type **operator\|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.602.1 Detailed Description

template<>struct std::atomic< unsigned short >

Explicit specialization for unsigned short.

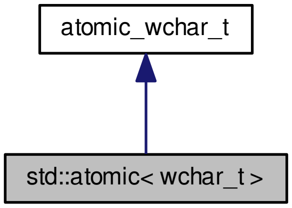
Definition at line 575 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.603 std::atomic< wchar_t > Struct Template Reference

Inheritance diagram for std::atomic< wchar_t >:



Public Types

- typedef atomic_wchar_t **__base_type**
- typedef wchar_t **__integral_type**

Public Member Functions

- **atomic** (const atomic &)=delete
- constexpr **atomic** (__integral_type __i) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m1, memory_order __m2) volatilenoexcept

- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__int_type &__i1, __int_type __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **exchange** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_add** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_and** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_or** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_sub** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- __int_type **fetch_xor** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __int_type **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator __int_type** () const noexcept
- **operator __int_type** () const volatilenoexcept
- __int_type **operator&=** (__int_type __i) noexcept
- __int_type **operator&=** (__int_type __i) volatilenoexcept
- __int_type **operator++** (int) noexcept
- __int_type **operator++** (int) volatilenoexcept
- __int_type **operator++** () noexcept
- __int_type **operator++** () volatilenoexcept
- __int_type **operator+=** (__int_type __i) noexcept
- __int_type **operator+=** (__int_type __i) volatilenoexcept
- __int_type **operator--** (int) noexcept
- __int_type **operator--** (int) volatilenoexcept
- __int_type **operator--** () noexcept
- __int_type **operator--** () volatilenoexcept
- __int_type **operator-=** (__int_type __i) noexcept
- __int_type **operator-=** (__int_type __i) volatilenoexcept
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &) volatile=delete
- __int_type **operator^=** (__int_type __i) noexcept
- __int_type **operator^=** (__int_type __i) volatilenoexcept
- __int_type **operator|=** (__int_type __i) noexcept
- __int_type **operator|=** (__int_type __i) volatilenoexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__int_type __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.603.1 Detailed Description

template<>struct std::atomic< wchar_t >

Explicit specialization for wchar_t.

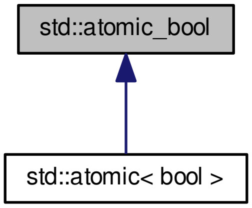
Definition at line 708 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.604 std::atomic_bool Struct Reference

Inheritance diagram for std::atomic_bool:



Public Member Functions

- **atomic_bool** (const atomic_bool &)=delete
- constexpr **atomic_bool** (bool __i) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) volatilenoexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst) volatilenoexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatilenoexcept
- bool **load** (memory_order __m=memory_order_seq_cst) const noexcept

- bool **load** (memory_order __m=memory_order_seq_cst) const volatilenoexcept
- **operator bool** () const noexcept
- **operator bool** () const volatilenoexcept
- atomic_bool & **operator=** (const atomic_bool &)=delete
- atomic_bool & **operator=** (const atomic_bool &) volatile=delete
- bool **operator=** (bool __i) noexcept
- bool **operator=** (bool __i) volatilenoexcept
- void **store** (bool __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (bool __i, memory_order __m=memory_order_seq_cst) volatilenoexcept

4.604.1 Detailed Description

atomic_bool

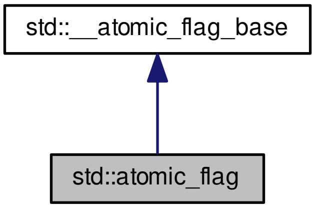
Definition at line 54 of file atomic.

The documentation for this struct was generated from the following file:

- atomic

4.605 std::atomic_flag Struct Reference

Inheritance diagram for std::atomic_flag:



Public Member Functions

- **atomic_flag** (const atomic_flag &)=delete
- constexpr **atomic_flag** (bool __i) noexcept
- void **clear** (memory_order __m=memory_order_seq_cst) noexcept
- void **clear** (memory_order __m=memory_order_seq_cst) volatilenoexcept
- atomic_flag & **operator=** (const atomic_flag &)=delete
- atomic_flag & **operator=** (const atomic_flag &) volatile=delete
- bool **test_and_set** (memory_order __m=memory_order_seq_cst) noexcept
- bool **test_and_set** (memory_order __m=memory_order_seq_cst) volatilenoexcept

Public Attributes

- __atomic_flag_data_type **_M_i**

4.605.1 Detailed Description

atomic_flag

Definition at line 271 of file atomic_base.h.

The documentation for this struct was generated from the following file:

- atomic_base.h

4.606 std::auto_ptr< _Tp > Class Template Reference

Public Types

- typedef _Tp element_type

Public Member Functions

- auto_ptr (element_type ∗__p=0) throw ()
- auto_ptr (auto_ptr &__a) throw ()
- template<typename _Tp1 >
  auto_ptr (auto_ptr< _Tp1 > &__a) throw ()
- auto_ptr (auto_ptr_ref< element_type > __ref) throw ()
- ∼auto_ptr ()
- element_type ∗ get () const throw ()
- template<typename _Tp1 >
  operator auto_ptr< _Tp1 > () throw ()
- template<typename _Tp1 >
  operator auto_ptr_ref< _Tp1 > () throw ()
- element_type & operator∗ () const throw ()
- element_type ∗ operator-> () const throw ()
- auto_ptr & operator= (auto_ptr &__a) throw ()
- template<typename _Tp1 >
  auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()
- auto_ptr & **operator=** (auto_ptr_ref< element_type > __ref) throw ()
- element_type ∗ release () throw ()
- void reset (element_type ∗__p=0) throw ()

4.606.1 Detailed Description

template<typename _Tp>class std::auto_ptr< _Tp >

A simple smart pointer providing strict ownership semantics.

The Standard says:

```
An  auto_ptr owns the object it holds a pointer to.  Copying
an  auto_ptr copies the pointer and transfers ownership to the
destination.  If more than one  auto_ptr owns the same object
at the same time the behavior of the program is undefined.


The uses of  auto_ptr include providing temporary
exception-safety for dynamically allocated memory, passing
ownership of dynamically allocated memory to a function, and
returning dynamically allocated memory from a function.   auto_ptr does not meet the CopyConstructibl
requirements for Standard Library  container elements and thus
instantiating a Standard Library container with an  auto_ptr results in undefined behavior.
```

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with auto_ptr can be found in the libstdc++ testsuite.

_GLIBCXX_RESOLVE_LIB_DEFECTS 127. auto_ptr<> conversion issues These resolutions have all been incorporated.

Definition at line 87 of file auto_ptr.h.

4.606.2 Member Typedef Documentation

4.606.2.1 template<typename _Tp> typedef _Tp **std::auto_ptr< _Tp >::element_type**

The pointed-to type.

Definition at line 94 of file auto_ptr.h.

4.606.3 Constructor & Destructor Documentation

4.606.3.1 template<typename _Tp> **std::auto_ptr< _Tp >::auto_ptr (element_type ∗ *__p* = 0) throw ()** `[inline]`, `[explicit]`

An auto_ptr is usually constructed from a raw pointer.

Parameters

__p	A pointer (defaults to NULL).

This object now *owns* the object pointed to by *__p*.

Definition at line 103 of file auto_ptr.h.

4.606.3.2 template<typename _Tp> **std::auto_ptr< _Tp >::auto_ptr (auto_ptr< _Tp > & *__a*) throw ()** `[inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

__a	Another auto_ptr of the same type.

This object now *owns* the object previously owned by *__a*, which has given up ownership.

Definition at line 112 of file auto_ptr.h.

4.606.3.3 template<typename _Tp> template<typename _Tp1 > std::auto_ptr< _Tp >::auto_ptr (auto_ptr< _Tp1 > & *__a*) throw () `[inline]`

An auto_ptr can be constructed from another auto_ptr.

Parameters

__a	Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by __*a*, which has given up ownership.

Definition at line 125 of file auto_ptr.h.

4.606.3.4 template<typename _Tp> std::auto_ptr< _Tp >::~auto_ptr ( ) `[inline]`

When the auto_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get()` is `NULL`), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if _Tp::~_Tp() throws, but this is prohibited. [17.4.3.6]/2

Definition at line 170 of file auto_ptr.h.

4.606.3.5 template<typename _Tp> std::auto_ptr< _Tp >::auto_ptr (auto_ptr_ref< element_type > *__ref*) throw () `[inline]`

Automatic conversions.

These operations convert an auto_ptr into and from an auto_ptr_ref automatically as needed. This allows constructs such as

```
auto_ptr<Derived>  func_returning_auto_ptr(.....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(.....);
```

Definition at line 260 of file auto_ptr.h.

4.606.4 Member Function Documentation

4.606.4.1 template<typename _Tp> element_type* std::auto_ptr< _Tp >::get (void) const throw () `[inline]`

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr still owns the memory.

Definition at line 211 of file auto_ptr.h.

4.606.4.2 template<typename _Tp> element_type& std::auto_ptr< _Tp >::operator* ( ) const throw () [inline]

Smart pointer dereferencing.

If this auto_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 181 of file auto_ptr.h.

4.606.4.3 template<typename _Tp> element_type* std::auto_ptr< _Tp >::operator-> ( ) const throw () [inline]

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.

Definition at line 194 of file auto_ptr.h.

4.606.4.4 template<typename _Tp> auto_ptr& std::auto_ptr< _Tp >::operator= (auto_ptr< _Tp > & __a) throw () [inline]

auto_ptr assignment operator.

Parameters

__a	Another auto_ptr of the same type.

This object now *owns* the object previously owned by *__a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 136 of file auto_ptr.h.

References std::auto_ptr< _Tp >::reset().

4.606.4.5 template<typename _Tp> template<typename _Tp1 > auto_ptr& std::auto_ptr< _Tp >::operator= (auto_ptr< _Tp1 > & __a) throw () [inline]

auto_ptr assignment operator.

Parameters

__a	Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.

This object now *owns* the object previously owned by *__a*, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 154 of file auto_ptr.h.

References std::auto_ptr< _Tp >::reset().

4.606.4.6 template<typename _Tp> element_type* std::auto_ptr< _Tp >::release ( ) throw () [inline]

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 225 of file auto_ptr.h.

4.606.4.7 template<typename _Tp> void **std::auto_ptr**< _Tp >::reset (**element_type** ∗ __p = 0) throw () [inline]

Forcibly deletes the managed object.

Parameters

__p	A pointer (defaults to NULL).

This object now *owns* the object pointed to by *__p*. The previous object has been deleted.

Definition at line 240 of file auto_ptr.h.

Referenced by std::auto_ptr< _Tp >::operator=().

The documentation for this class was generated from the following file:

- auto_ptr.h

4.607 std::auto_ptr_ref< _Tp1 > Struct Template Reference

Public Member Functions

- **auto_ptr_ref** (_Tp1 ∗__p)

Public Attributes

- _Tp1 ∗ **_M_ptr**

4.607.1 Detailed Description

template<typename _Tp1>struct std::auto_ptr_ref< _Tp1 >

A wrapper class to provide auto_ptr with reference semantics. For example, an auto_ptr can be assigned (or constructed from) the result of a function which returns an auto_ptr by value.

All the auto_ptr_ref stuff should happen behind the scenes.

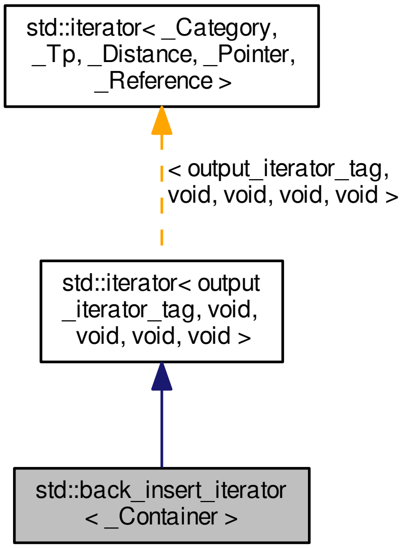
Definition at line 48 of file auto_ptr.h.

The documentation for this struct was generated from the following file:

- auto_ptr.h

4.608 std::back_insert_iterator< _Container > Class Template Reference

Inheritance diagram for std::back_insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- back_insert_iterator (_Container &__x)
- back_insert_iterator & operator∗ ()
- back_insert_iterator & operator++ ()
- back_insert_iterator operator++ (int)
- back_insert_iterator & operator= (const typename _Container::value_type &__value)
- back_insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container ∗ **container**

4.608.1 Detailed Description

template<typename _Container>class std::back_insert_iterator< _Container >

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using push_back.

Tip: Using the back_inserter function to create these iterators can save typing.

Definition at line 402 of file stl_iterator.h.

4.608.2 Member Typedef Documentation

4.608.2.1 template<typename _Container > typedef _Container **std::back_insert_iterator**< _Container >**::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 410 of file stl_iterator.h.

4.608.2.2 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::difference_type** `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.608.2.3 typedef **output_iterator_tag std::iterator**< **output_iterator_tag** , void , void , void , void >**::iterator_category** `[inherited]`

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.608.2.4 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::pointer** `[inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.608.2.5 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::reference** `[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.608.2.6 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::value_type** `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.608.3 Constructor & Destructor Documentation

4.608.3.1 template<typename _Container > **std::back_insert_iterator**< _Container >**::back_insert_iterator** (_Container & *__x*) `[inline]`, `[explicit]`

The only way to create this iterator is with a container.

Definition at line 414 of file stl_iterator.h.

4.608.4 Member Function Documentation

4.608.4.1 template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator* ( ) [inline]

Simply returns *this.

Definition at line 452 of file stl_iterator.h.

4.608.4.2 template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator++ ( ) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 457 of file stl_iterator.h.

4.608.4.3 template<typename _Container > back_insert_iterator std::back_insert_iterator< _Container >::operator++ (int) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 462 of file stl_iterator.h.

4.608.4.4 template<typename _Container > back_insert_iterator& std::back_insert_iterator< _Container >::operator= (const typename _Container::value_type & __value) [inline]

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

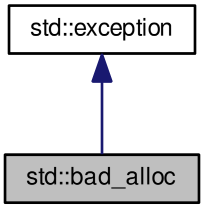
Definition at line 436 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- stl_iterator.h

4.609 std::bad_alloc Class Reference

Inheritance diagram for std::bad_alloc:



Public Member Functions

- virtual const char $*$ what () const throw ()

4.609.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 54 of file new.

4.609.2 Member Function Documentation

4.609.2.1 virtual const char$*$ std::bad_alloc::what ( ) const throw () `[virtual]`

Returns a C-style character string describing the general cause of the current error.

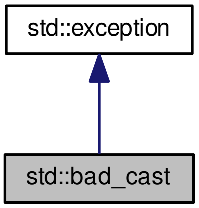
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- new

4.610 std::bad_cast Class Reference

Inheritance diagram for std::bad_cast:



Public Member Functions

- virtual const char $*$ what () const noexcept

4.610.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 187 of file typeinfo.

4.610.2 Member Function Documentation

4.610.2.1 virtual const char$*$ std::bad_cast::what ( ) const `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error.

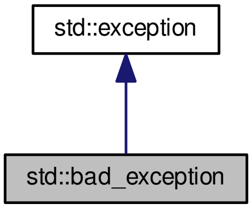
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- typeinfo

4.611 std::bad_exception Class Reference

Inheritance diagram for std::bad_exception:



Public Member Functions

- virtual const char $*$ what () const noexcept

4.611.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 73 of file exception.

4.611.2 Member Function Documentation

4.611.2.1 virtual const char$*$ std::bad_exception::what ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

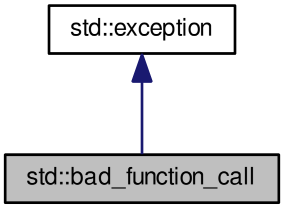
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- exception

4.612 std::bad_function_call Class Reference

Inheritance diagram for std::bad_function_call:



Public Member Functions

- const char ∗ what () const noexcept

4.612.1 Detailed Description

Exception class thrown when class template function's operator() is called with an empty target.

Definition at line 1767 of file functional.

4.612.2 Member Function Documentation

4.612.2.1 const char∗ std::bad_function_call::what ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

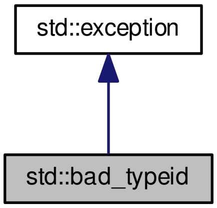
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- functional

4.613 std::bad_typeid Class Reference

Inheritance diagram for std::bad_typeid:



Public Member Functions

- virtual const char $*$ what () const noexcept

4.613.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

Definition at line 204 of file typeinfo.

4.613.2 Member Function Documentation

4.613.2.1 virtual const char$*$ std::bad_typeid::what ( ) const `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error.

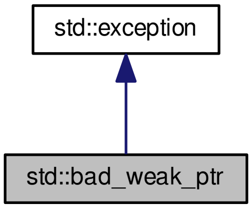
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- typeinfo

4.614 std::bad_weak_ptr Class Reference

Inheritance diagram for std::bad_weak_ptr:



Public Member Functions

- virtual char const $*$ what () const noexcept

4.614.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

Definition at line 64 of file shared_ptr_base.h.

4.614.2 Member Function Documentation

4.614.2.1 virtual char const$*$ std::bad_weak_ptr::what ( ) const `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error.

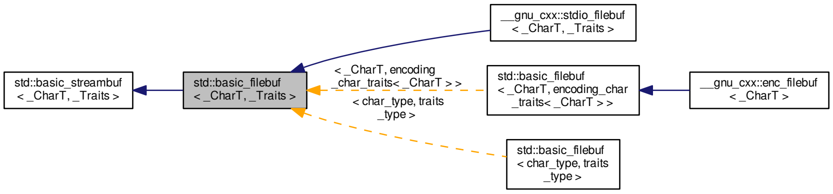
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- shared_ptr_base.h

4.615 std::basic_filebuf< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_filebuf< _CharT, _Traits >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf < char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf < char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- basic_filebuf ()
- virtual ∼basic_filebuf ()
- __filebuf_type ∗ close ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- __filebuf_type ∗ open (const char ∗__s, ios_base::openmode __mode)
- __filebuf_type ∗ open (const std::string &__s, ios_base::openmode __mode)
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type ∗__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type ∗__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf ∗ pubsetbuf (char_type ∗__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type ∗, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=_Traits::eof())
- virtual int_type pbackfail (int_type __c=_Traits::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type ∗ setbuf (char_type ∗__s, streamsize __n)
- void setg (char_type ∗__gbeg, char_type ∗__gnext, char_type ∗__gend)
- void setp (char_type ∗__pbeg, char_type ∗__pend)
- virtual streamsize showmanyc ()
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type ∗__s, streamsize __n)
- virtual streamsize xsputn (const char_type ∗__s, streamsize __n)

- char_type ∗ eback () const
- char_type ∗ gptr () const
- char_type ∗ egptr () const

- char_type ∗ pbase () const
- char_type ∗ pptr () const
- char_type ∗ epptr () const

Protected Attributes

- char_type ∗ _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type ∗ **_M_codecvt**
- char ∗ _M_ext_buf
- streamsize _M_ext_buf_size
- char ∗ **_M_ext_end**
- const char ∗ _M_ext_next
- __file_type **_M_file**
- char_type ∗ _M_in_beg
- char_type ∗ _M_in_cur
- char_type ∗ _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type ∗ _M_out_beg
- char_type ∗ _M_out_cur
- char_type ∗ _M_out_end
- bool _M_reading
- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type ∗ _M_pback_cur_save
- char_type ∗ _M_pback_end_save
- bool _M_pback_init

Friends

- class **ios_base**

4.615.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_filebuf< _CharT, _Traits >

The actual work of input and output (for files).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Requirements on traits_type, specific to this class:

- traits_type::pos_type must be fpos<traits_type::state_type>
- traits_type::off_type must be streamoff
- traits_type::state_type must be Assignable and DefaultConstructible,
- traits_type::state_type() must be the initial state for codecvt.

Definition at line 72 of file fstream.

4.615.2 Constructor & Destructor Documentation

4.615.2.1 template<typename _CharT , typename _Traits > **std::basic_filebuf**< _CharT, _Traits >::**basic_filebuf** ( )

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 79 of file fstream.tcc.

References std::basic_streambuf< _CharT, _Traits >::_M_buf_locale.

4.615.2.2 template<typename _CharT, typename _Traits> virtual **std::basic_filebuf**< _CharT, _Traits >::~**basic_filebuf** ( ) `[inline]`, `[virtual]`

The destructor closes the file first.

Definition at line 219 of file fstream.

4.615.3 Member Function Documentation

4.615.3.1 template<typename _CharT, typename _Traits> void **std::basic_filebuf**< _CharT, _Traits >::_M_create_pback ( ) `[inline]`, `[protected]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back

Definition at line 177 of file fstream.

4.615.3.2 template<typename _CharT, typename _Traits> void **std::basic_filebuf**< _CharT, _Traits >::_M_destroy_pback ( ) throw () `[inline]`, `[protected]`

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 194 of file fstream.

4.615.3.3 template<typename _CharT, typename _Traits> void **std::basic_filebuf**< _CharT, _Traits >::_M_set_buffer (**streamsize** ***__off***) `[inline]`, `[protected]`

This function sets the pointers of the internal buffer, both get and put areas. Typically:

__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 395 of file fstream.

4.615.3.4 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::**__filebuf_type** ∗ **std::basic_filebuf**< _CharT, _Traits >::close ( )

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 128 of file fstream.tcc.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), and std::basic_filebuf< char_type, traits_type >::∼basic_filebuf().

4.615.3.5 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::eback ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sungetc().

4.615.3.6 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::egptr ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.615.3.7 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.615.3.8 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::gbump (int __n) [inline], [protected], [inherited]

Moving the read position.

Parameters

__n	The delta by which to move.

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.615.3.9 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::getloc ( ) const [inline], [inherited]

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.615.3.10 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::gptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence

- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.615.3.11 template< typename _CharT , typename _Traits > void **std::basic_filebuf**< _CharT, _Traits >::imbue (const **locale** & *__loc*) `[protected]`, `[virtual]`

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 910 of file fstream.tcc.

References std::ios_base::cur.

4.615.3.12 template<typename _CharT, typename _Traits> **streamsize std::basic_streambuf**< _CharT, _Traits >::in_avail ( ) `[inline]`, `[inherited]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.615.3.13 template<typename _CharT, typename _Traits> bool **std::basic_filebuf**< _CharT, _Traits >::is_open ( ) const throw () `[inline]`

Returns true if the external file is open.

Definition at line 227 of file fstream.

Referenced by std::basic_ifstream< _CharT, _Traits >::is_open(), std::basic_ofstream< _CharT, _Traits >::is_open(), and std::basic_fstream< _CharT, _Traits >::is_open().

4.615.3.14 template< typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::**__filebuf_type** * **std::basic_filebuf**< _CharT, _Traits >::open (const char * *__s*, **ios_base::openmode** *__mode*)

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent fopen() flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596) +——————————————+ | ios_base Flag combination stdio equivalent | |binary in out trunc app | +——————————————+ | + w | | + + a | | + a | | + + w | | + r | | + + r+ | | + + + w+ | | + + + a+ | | + + a+ | +——————————————+ | + + wb | | + + + ab | | + + ab | | + + + wb | | + + rb | | + + + r+b | | + + + + w+b | | + + + + a+b | | + + + a+b | +——————————————+

Definition at line 94 of file fstream.tcc.

References std::ios_base::ate, std::ios_base::end, and std::basic_filebuf< _CharT, _Traits >::open().

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), and std::basic_fstream< _CharT, _Traits >::open().

4.615.3.15 template<typename _CharT, typename _Traits> __filebuf_type* std::basic_filebuf< _CharT, _Traits >::open (const std::string & *__s*, ios_base::openmode *__mode*) `[inline]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

Definition at line 280 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::open().

4.615.3.16 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (int_type *__c* = `_Traits::eof()`) `[protected]`, `[virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 422 of file fstream.tcc.

References std::ios_base::cur, and std::ios_base::out.

4.615.3.17 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (int_type *__c* = `_Traits::eof()`) `[protected]`, `[virtual]`

Tries to back up the input sequence.

Parameters

__c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 363 of file fstream.tcc.

References std::ios_base::cur, and std::ios_base::in.

4.615.3.18 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pbase ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence

- epptr() returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.615.3.19 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::pbump (int __n) [inline], [protected], [inherited]

Moving the write position.

Parameters

__n	The delta by which to move.

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.615.3.20 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.615.3.21 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale & __loc) [inline], [inherited]

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

Definition at line 209 of file streambuf.

4.615.3.22 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type *__off*, ios_base::seekdir *__way*, ios_base::openmode *__mode* = ios_base::in \| ios_base::out) [inline], [inherited]

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.615.3.23 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type *__sp*, ios_base::openmode *__mode* = ios_base::in \| ios_base::out) [inline], [inherited]

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

4.615.3.24 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (char_type * *__s*, streamsize *__n*) [inline], [inherited]

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

4.615.3.25 template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::sync().

4.615.3.26 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sbumpc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::istreambuf_iterator< _CharT, _Traits >::operator++(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.615.3.27 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::pos_type **std::basic_filebuf**< _CharT, _Traits >::seekoff (off_type , **ios_base::seekdir** , **ios_base::openmode** = **ios_base::in** | **ios_base::out**) `[protected]`, `[virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 713 of file fstream.tcc.

References std::ios_base::cur.

4.615.3.28 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::pos_type **std::basic_filebuf**< _CharT, _Traits >::seekpos (pos_type , **ios_base::openmode** = **ios_base::in** | **ios_base::out**) `[protected]`, `[virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 773 of file fstream.tcc.

References std::ios_base::beg.

4.615.3.29 template<typename _CharT , typename _Traits > **basic_filebuf**< _CharT, _Traits >::**__streambuf_type** * **std::basic_filebuf**< _CharT, _Traits >::setbuf (char_type * ***__s***, **streamsize** ***__n***) `[protected]`, `[virtual]`

Manipulates the buffer.

Parameters

__s	Pointer to a buffer area.
__n	Size of *__s*.

Returns

`this`

If no file has been opened, and both *__s* and *__n* are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 684 of file fstream.tcc.

4.615.3.30 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setg (**char_type** ∗ *__gbeg,* **char_type** ∗ *__gnext,* **char_type** ∗ *__gend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Definition at line 509 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.615.3.31 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setp (**char_type** ∗ *__pbeg,* **char_type** ∗ *__pend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Definition at line 555 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.615.3.32 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( ) [inline], [inherited]

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.615.3.33 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s, streamsize __n) [inline], [inherited]

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill *__s*[0] through *__s*[__n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.615.3.34 template<typename _CharT , typename _Traits > streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ( ) [protected], [virtual]

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.
The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return `eof()` but that they will return immediately.*

> The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 178 of file fstream.tcc.

References std::ios_base::binary, and std::ios_base::in.

4.615.3.35 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::snextc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

> The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.615.3.36 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sputbackc (**char_type** ***__c***) `[inline]`, `[inherited]`

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

> The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.

Definition at line 372 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::putback().

4.615.3.37 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sputc (**char_type** ***__c***) `[inline]`, `[inherited]`

Entry point for all single-character output functions.

Parameters

__c	A character to output.

Returns

> *__c*, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *__c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 424 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), and std::ostreambuf_iterator< _CharT, _Traits >::operator=().

4.615.3.38 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Entry point for all single-character output functions.

Parameters

__s	A buffer read area.
__n	A count.

One of two public output functions.

Returns xsputn(__s,__n). The effect is to write *__s*[0] through *__s*[__n-1] to the output sequence, if possible.

Definition at line 450 of file streambuf.

4.615.3.39 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) `[inline]`, `[inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

4.615.3.40 template<typename _CharT , typename _Traits > int std::basic_filebuf< _CharT, _Traits >::sync ( ) `[protected]`, `[virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 893 of file fstream.tcc.

4.615.3.41 template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( ) [inline], [protected], [virtual], [inherited]

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 700 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc().

4.615.3.42 template<typename _CharT , typename _Traits > basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ( ) [protected], [virtual]

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html`

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 204 of file fstream.tcc.

References std::ios_base::in, and std::min().

4.615.3.43 template<typename _CharT , typename _Traits > streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (char_type * __s, streamsize __n) [protected], [virtual]

Multiple character extraction.

Parameters

__s	A buffer area.
__n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 549 of file fstream.tcc.

References std::ios_base::in.

4.615.3.44 template<typename _CharT , typename _Traits > **streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (const char_type * __s, streamsize __n)** `[protected]`, `[virtual]`

Multiple character insertion.

Parameters

__*s*	A buffer area.
__*n*	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 637 of file fstream.tcc.

References std::min(), and std::ios_base::out.

4.615.4 Member Data Documentation

4.615.4.1 template<typename _CharT, typename _Traits> char_type* **std::basic_filebuf< _CharT, _Traits >::_M_buf** `[protected]`

Pointer to the beginning of internal buffer.

Definition at line 114 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.615.4.2 template<typename _CharT, typename _Traits> **locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale** `[protected]`, `[inherited]`

Current locale setting.

Definition at line 192 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf(), std::basic_streambuf< char_type, traits_type >::getloc(), and std::basic_streambuf< char_type, traits_type >::pubimbue().

4.615.4.3 template<typename _CharT, typename _Traits> size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size [protected]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 121 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.615.4.4 template<typename _CharT, typename _Traits> char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().

Definition at line 156 of file fstream.

4.615.4.5 template<typename _CharT, typename _Traits> streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected]

Size of buffer held by _M_ext_buf.

Definition at line 161 of file fstream.

4.615.4.6 template<typename _CharT, typename _Traits> const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected]

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().

Definition at line 168 of file fstream.

4.615.4.7 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]

Start of get area.

Definition at line 184 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::eback(), and std::basic_streambuf< char_type, traits_type >::setg().

4.615.4.8 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]

Current read area.

Definition at line 185 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::gbump(), std::basic_streambuf< char_type, traits_type >::gptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.615.4.9 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]

End of get area.

Definition at line 186 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::egptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.615.4.10 template<typename _CharT, typename _Traits> ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode [protected]

Place to stash in || out || in | out settings for current filebuf.

Definition at line 99 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.615.4.11 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]

Start of put area.

Definition at line 187 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbase(), and std::basic_streambuf< char_type, traits_type >::setp().

4.615.4.12 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]

Current put area.

Definition at line 188 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbump(), std::basic_streambuf< char_type, traits_type >::pptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.615.4.13 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]

End of put area.

Definition at line 189 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::epptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.615.4.14 template<typename _CharT, typename _Traits> char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 142 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback().

4.615.4.15 template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 143 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), and std::basic_filebuf< char_type, traits_type >::_M_destroy_pback().

4.615.4.16 template<typename _CharT, typename _Traits> char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 144 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), and std::basic_filebuf< char_type, traits_type >::_M_destroy_pback().

4.615.4.17 template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init [protected]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Definition at line 145 of file fstream.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), and std::basic_filebuf< char_type, traits_type >::_M_destroy_pback().

4.615.4.18 template<typename _CharT, typename _Traits> bool std::basic_filebuf< _CharT, _Traits >::_M_reading [protected]

_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;

NB: _M_reading == true && _M_writing == true is unused.

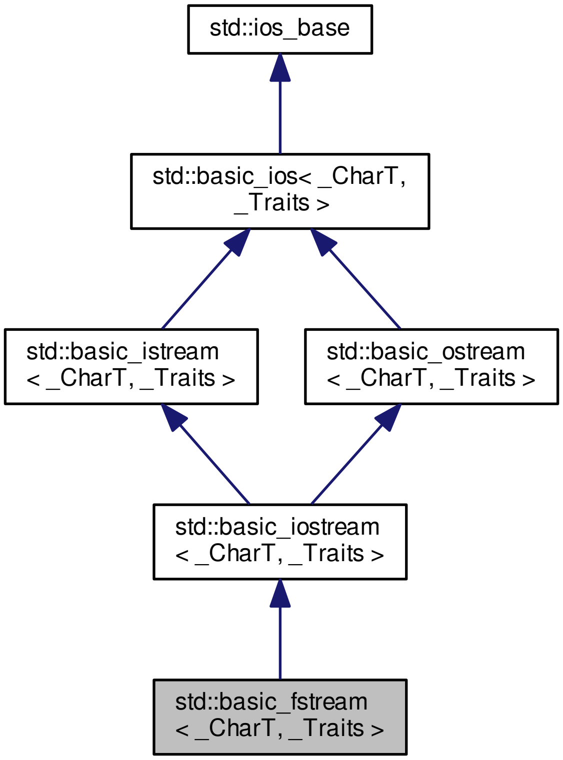
Definition at line 133 of file fstream.

The documentation for this class was generated from the following files:

- fstream
- fstream.tcc

4.616 std::basic_fstream< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_fstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf
  < char_type, traits_type > **__filebuf_type**
- typedef basic_ios< char_type,
  traits_type > **__ios_type**
- typedef basic_iostream
  < char_type, traits_type > **__iostream_type**
- typedef basic_istream< _CharT,
  _Traits > **__istream_type**
- typedef num_get< _CharT,
  istreambuf_iterator< _CharT,
  _Traits > > **__num_get_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_fstream ()
- basic_fstream (const char ∗__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- basic_fstream (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- ∼basic_fstream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()

- streamsize gcount () const
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- void open (const char ∗__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- __ostream_type & operator<< (const void ∗__p)
- __ostream_type & operator<< (__streambuf_type ∗__sb)
- __istream_type & operator>> (void ∗&__p)
- __istream_type & operator>> (__streambuf_type ∗__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- __filebuf_type ∗ rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (pos_type)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(∗__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(∗__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(∗__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)

  - __istream_type & operator>> (unsigned int &__n)
  - __istream_type & operator>> (long &__n)
  - __istream_type & operator>> (unsigned long &__n)
  - __istream_type & operator>> (long long &__n)
  - __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - int_type get ()
  - __istream_type & get (char_type &__c)
  - __istream_type & get (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & get (char_type ∗__s, streamsize __n)
  - __istream_type & get (__streambuf_type &__sb, char_type __delim)
  - __istream_type & get (__streambuf_type &__sb)
  - __istream_type & getline (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & getline (char_type ∗__s, streamsize __n)
  - __istream_type & ignore (streamsize __n, int_type __delim)
  - __istream_type & ignore (streamsize __n)
  - __istream_type & ignore ()
  - int_type peek ()
  - __istream_type & read (char_type ∗__s, streamsize __n)
  - streamsize readsome (char_type ∗__s, streamsize __n)
  - __istream_type & putback (char_type __c)
  - __istream_type & unget ()
  - int sync ()
  - pos_type tellg ()
  - __istream_type & seekg (pos_type)
  - __istream_type & seekg (off_type, ios_base::seekdir)

- operator void ∗ () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(∗__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(∗__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(∗__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __ostream_type & operator<< (long __n)
  - __ostream_type & operator<< (unsigned long __n)
  - __ostream_type & operator<< (bool __n)
  - __ostream_type & operator<< (short __n)
  - __ostream_type & operator<< (unsigned short __n)
  - __ostream_type & operator<< (int __n)
  - __ostream_type & operator<< (unsigned int __n)
  - __ostream_type & operator<< (long long __n)
  - __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

  - __ostream_type & put (char_type __c)
  - void _M_write (const char_type ∗__s, streamsize __n)
  - __ostream_type & write (const char_type ∗__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in

- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
  __ostream_type & **_M_insert** (_ValueT __v)
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**

- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.616.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_fstream< _CharT, _Traits >

Controlling input and output for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from and writing to named files, using the inherited functions from std::basic_iostream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

Definition at line 776 of file fstream.

4.616.2 Member Typedef Documentation

4.616.2.1 template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type `[inherited]`

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.616.2.2 typedef void(∗ std::ios_base::event_callback)(event __e, ios_base &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.616.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed

- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.616.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.616.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in

- out
- trunc

Definition at line 361 of file ios_base.h.

4.616.2.6 typedef _Ios_Seekdir std::ios_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.616.3 Member Enumeration Documentation

4.616.3.1 enum std::ios_base::event [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.616.4 Constructor & Destructor Documentation

4.616.4.1 template<typename _CharT , typename _Traits > std::basic_fstream< _CharT, _Traits >::basic_fstream ( ) [inline]

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 803 of file fstream.

References std::basic_ios< _CharT, _Traits >::init().

4.616.4.2 template<typename _CharT , typename _Traits > std::basic_fstream< _CharT, _Traits >::basic_fstream (const char ∗ __s, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]

Create an input/output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

Tip: When using std::string to hold the filename, you must use .c_str() before passing it to this constructor.

Definition at line 816 of file fstream.

References std::basic_ios< _CharT, _Traits >::init(), and std::basic_fstream< _CharT, _Traits >::open().

4.616.4.3 template<typename _CharT , typename _Traits > std::basic_fstream< _CharT, _Traits >::basic_fstream (const std::string & __s, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]

Create an input/output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

Definition at line 831 of file fstream.

References std::basic_ios< _CharT, _Traits >::init(), and std::basic_fstream< _CharT, _Traits >::open().

4.616.4.4 template<typename _CharT , typename _Traits > std::basic_fstream< _CharT, _Traits >::~basic_fstream ( ) [inline]

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 846 of file fstream.

4.616.5 Member Function Documentation

4.616.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.616.5.2 template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::_M_write (const char_type * __s, streamsize __n) [inline], [inherited]

Core write functionality, without sentry.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Definition at line 311 of file ostream.

Referenced by std::basic_ostream< _CharT, _Traits >::write().

4.616.5.3 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.616.5.4 template<typename _CharT , typename _Traits > void **basic_ios**< _CharT, _Traits >::clear (**iostate** *__state* = **goodbit**) `[inherited]`

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.616.5.5 template<typename _CharT , typename _Traits > void **std::basic_fstream**< _CharT, _Traits >::close ( ) `[inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 926 of file fstream.

References std::basic_filebuf< _CharT, _Traits >::close(), std::ios_base::failbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.6 template<typename _CharT , typename _Traits > **basic_ios**< _CharT, _Traits > & **basic_ios**< _CharT, _Traits >::copyfmt (const **basic_ios**< _CharT, _Traits > & *__rhs*) `[inherited]`

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.616.5.7 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.616.5.8 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.616.5.9 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate *__except*) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.616.5.10 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::fail ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.616.5.11 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill ( ) const `[inline]`, `[inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.616.5.12 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill (**char_type** *__ch*) `[inline]`, `[inherited]`

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.616.5.13 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.616.5.14 fmtflags std::ios_base::flags (fmtflags *__fmtfl*) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.616.5.15 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::flush ( ) [inherited]

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::flush().

4.616.5.16 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.616.5.17 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.18 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type & __c) [inherited]

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

∗this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.19 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type ∗ __s, streamsize __n, char_type __delim) [inherited]

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.616.5.20 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type ∗ *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

Definition at line 354 of file istream.

4.616.5.21 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*, char_type *__delim*) `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

4.616.5.22 template<typename _CharT, typename _Traits> **__istream_type& std::basic_istream**< _CharT, _Traits >::get (**__streambuf_type** & *__sb*) `[inline]`, `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

∗this

Returns `get`(__sb,widen('\n')).

Definition at line 387 of file istream.

4.616.5.23 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::getline (char_type ∗ *__s*, **streamsize** *__n*, char_type *__delim*) `[inherited]`

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

∗this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.616.5.24 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * *__s*, streamsize *__n*) `[inline]`, `[inherited]`

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

Definition at line 427 of file istream.

Referenced by std::basic_istream< char >::getline().

4.616.5.25 locale std::ios_base::getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.616.5.26 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.616.5.27 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize __n, int_type __delim) [inherited]

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

∗this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.616.5.28 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize __n) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.616.5.29 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.30 template<typename _CharT , typename _Traits > **locale basic_ios**< _CharT, _Traits >::imbue (const **locale** & *__loc*) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.616.5.31 template<typename _CharT, typename _Traits> void **basic_ios**< _CharT, _Traits >::init (**basic_streambuf**< _CharT, _Traits > * *__sb*) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.616.5.32 template<typename _CharT , typename _Traits > bool **std::basic_fstream**< _CharT, _Traits >::is_open ( ) [inline]

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

Definition at line 865 of file fstream.

References std::basic_filebuf< _CharT, _Traits >::is_open().

4.616.5.33 long& std::ios_base::iword (int __ix) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.616.5.34 template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow (char_type __c, char __dfault) const [inline], [inherited]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 424 of file basic_ios.h.

4.616.5.35 template<typename _CharT , typename _Traits > void std::basic_fstream< _CharT, _Traits >::open (const char * __s, ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

Tip: When using std::string to hold the filename, you must use .c_str() before passing it to this constructor.

Definition at line 886 of file fstream.

References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::failbit, std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream().

4.616.5.36 template<typename _CharT , typename _Traits > void **std::basic_fstream**< _CharT, _Traits >::open (const **std::string** & *__s*, **ios_base::openmode** *__mode* = **ios_base::in** | **ios_base::out**) `[inline]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 907 of file fstream.

References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::failbit, std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.37 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void * ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.616.5.38 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.616.5.39 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ostream_type &(*)(__ostream_type &)** *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 108 of file ostream.

4.616.5.40 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 117 of file ostream.

4.616.5.41 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 127 of file ostream.

4.616.5.42 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

4.616.5.43 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.616.5.44 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.616.5.45 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (short *__n*) `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.616.5.46 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short *__n*) `[inline]`, `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.616.5.47 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (int *__n*) `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.616.5.48 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

4.616.5.49 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

4.616.5.50 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

4.616.5.51 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

4.616.5.52 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

4.616.5.53 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.616.5.54 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * __p) [inline], [inherited]

Pointer arithmetic inserters.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.616.5.55 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.56 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 120 of file istream.

4.616.5.57 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 124 of file istream.

4.616.5.58 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 131 of file istream.

4.616.5.59 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file istream.

4.616.5.60 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (short & __n) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.61 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

4.616.5.62 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (int & __n) `[inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.63 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) `[inline]`, `[inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file istream.

4.616.5.64 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) `[inline]`, `[inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file istream.

4.616.5.65 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file istream.

4.616.5.66 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file istream.

4.616.5.67 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file istream.

4.616.5.68 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file istream.

4.616.5.69 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file istream.

4.616.5.70 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file istream.

4.616.5.71 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& __p) [inline], [inherited]

Basic arithmetic extractors.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file istream.

4.616.5.72 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.73 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::peek (void) [inherited]

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.74 **streamsize** std::ios_base::precision ( ) const `[inline]`, `[inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.616.5.75 **streamsize** std::ios_base::precision (**streamsize** *__prec*) `[inline]`, `[inherited]`

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.616.5.76 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::put (char_type *__c*) `[inherited]`

Simple insertion.

Parameters

__c	The character to insert.

Returns

∗this

Tries to insert `__c`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.616.5.77 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::putback (char_type *__c*) `[inherited]`

Unextracting a single character.

Parameters

__c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

Referenced by std::operator>>().

4.616.5.78 void∗& std::ios_base::pword (int *__ix*) `[inline]`, `[inherited]`

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.616.5.79 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > ∗ basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > ∗ *__sb*) `[inherited]`

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.616.5.80 template<typename _CharT , typename _Traits > __filebuf_type* std::basic_fstream< _CharT, _Traits >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

Definition at line 857 of file fstream.

4.616.5.81 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.616.5.82 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::read (char_type * __s, streamsize __n) [inherited]

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

∗this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.83 template<typename _CharT , typename _Traits > streamsize basic_istream< _CharT, _Traits >::readsome (char_type ∗ *__s,* streamsize *__n*) `[inherited]`

Extraction until the buffer is exhausted, but no more.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A` == `-1`, sets eofbit and extracts no characters
- if `A` == `0`, extracts no characters
- if `A` > `0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.84 void std::ios_base::register_callback (event_callback *__fn,* int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.616.5.85 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (pos_type *__pos*) `[inherited]`

Changing the current read position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.86 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (off_type *__off*, **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.87 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (pos_type *__pos*) `[inherited]`

Changing the current write position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.88 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (off_type *__off,* **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.89 **fmtflags** std::ios_base::setf (**fmtflags** *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.616.5.90 fmtflags std::ios_base::setf (fmtflags __fmtfl, fmtflags __mask) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.616.5.91 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate __state) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.616.5.92 template<typename _CharT , typename _Traits > int basic_istream< _CharT, _Traits >::sync (void) `[inherited]`

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.616.5.93 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.616.5.94 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg (void) `[inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.616.5.95 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp ( ) `[inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.

Definition at line 237 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.616.5.96 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.616.5.97 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.616.5.98 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::unget (void) [inherited]

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().

4.616.5.99 void std::ios_base::unsetf (**fmtflags** *__mask*) `[inline], [inherited]`

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.616.5.100 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::widen (char *__c*) const `[inline], [inherited]`

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.616.5.101 **streamsize** std::ios_base::width ( ) const `[inline], [inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.616.5.102 streamsize std::ios_base::width (streamsize __wide) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.616.5.103 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]

Character string insertion.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Returns

∗this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic_ostream< _CharT, _Traits >::_M_write(), and std::ios_base::badbit.

4.616.5.104 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.616.6 Member Data Documentation

4.616.6.1 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic_istream< char >::gcount(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::unget(), and std::basic_istream< char >::~basic_istream().

4.616.6.2 const fmtflags std::ios_base::adjustfield [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.616.6.3 const openmode std::ios_base::app [static], [inherited]

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.616.6.4 const openmode std::ios_base::ate [static], [inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.616.6.5 const iostate std::ios_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.616.6.6 const fmtflags std::ios_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream<

_CharT, _Traits >::operator<<().

4.616.6.7 const seekdir std::ios_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.616.6.8 const openmode std::ios_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.616.6.9 const fmtflags std::ios_base::boolalpha [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.616.6.10 const seekdir std::ios_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.616.6.11 const fmtflags std::ios_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.616.6.12 const seekdir std::ios_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.616.6.13 const iostate std::ios_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time-

_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.616.6.14 const iostate std::ios_base::failbit `[static]`, `[inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.616.6.15 const fmtflags std::ios_base::fixed `[static]`, `[inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.616.6.16 const fmtflags std::ios_base::floatfield `[static]`, `[inherited]`

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.616.6.17 const iostate std::ios_base::goodbit `[static]`, `[inherited]`

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.616.6.18 const fmtflags std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.616.6.19 const openmode std::ios_base::in [static], [inherited]

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.616.6.20 const fmtflags std::ios_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.616.6.21 const fmtflags std::ios_base::left [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.616.6.22 const fmtflags std::ios_base::oct [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.616.6.23 const openmode std::ios_base::out [static], [inherited]

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.616.6.24 const fmtflags std::ios_base::right [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.616.6.25 const **fmtflags** std::ios_base::scientific `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.616.6.26 const **fmtflags** std::ios_base::showbase `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.616.6.27 const **fmtflags** std::ios_base::showpoint `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.616.6.28 const **fmtflags** std::ios_base::showpos `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.616.6.29 const **fmtflags** std::ios_base::skipws `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.616.6.30 const **openmode** std::ios_base::trunc `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.616.6.31 const **fmtflags** std::ios_base::unitbuf `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.616.6.32 const **fmtflags** std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

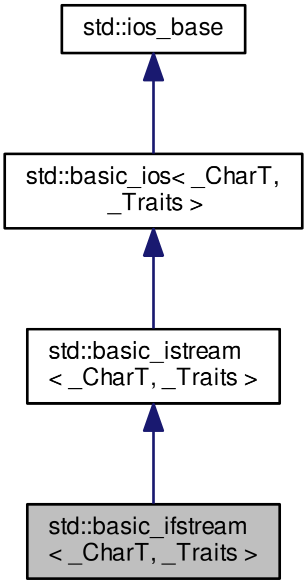
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- fstream

4.617 std::basic_ifstream< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_ifstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf
  < char_type, traits_type > **__filebuf_type**
- typedef basic_ios< _CharT,
  _Traits > **__ios_type**
- typedef basic_istream
  < char_type, traits_type > **__istream_type**
- typedef num_get< _CharT,
  istreambuf_iterator< _CharT,
  _Traits > > **__num_get_type**
- typedef basic_streambuf
  < _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)

- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_ifstream ()
- basic_ifstream (const char *__s, ios_base::openmode __mode=ios_base::in)
- basic_ifstream (const std::string &__s, ios_base::openmode __mode=ios_base::in)
- ∼basic_ifstream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- void open (const char *__s, ios_base::openmode __mode=ios_base::in)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::in)
- __istream_type & operator>> (void *&__p)
- __istream_type & operator>> (__streambuf_type *__sb)

- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- __filebuf_type ∗ rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(∗__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(∗__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(∗__pf)(ios_base &))

Extractors

All the `operator>>` *functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __istream_type & operator>> (bool &__n)
  - __istream_type & operator>> (short &__n)
  - __istream_type & operator>> (unsigned short &__n)
  - __istream_type & operator>> (int &__n)
  - __istream_type & operator>> (unsigned int &__n)
  - __istream_type & operator>> (long &__n)
  - __istream_type & operator>> (unsigned long &__n)
  - __istream_type & operator>> (long long &__n)
  - __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - int_type get ()
  - __istream_type & get (char_type &__c)
  - __istream_type & get (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & get (char_type ∗__s, streamsize __n)
  - __istream_type & get (__streambuf_type &__sb, char_type __delim)
  - __istream_type & get (__streambuf_type &__sb)
  - __istream_type & getline (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & getline (char_type ∗__s, streamsize __n)
  - __istream_type & ignore (streamsize __n, int_type __delim)
  - __istream_type & ignore (streamsize __n)
  - __istream_type & ignore ()
  - int_type peek ()
  - __istream_type & read (char_type ∗__s, streamsize __n)
  - streamsize readsome (char_type ∗__s, streamsize __n)
  - __istream_type & putback (char_type __c)
  - __istream_type & unget ()
  - int sync ()
  - pos_type tellg ()
  - __istream_type & seekg (pos_type)
  - __istream_type & seekg (off_type, ios_base::seekdir)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out

- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT,
  _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.617.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_ifstream< _CharT, _Traits >

Controlling input for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from named files, using the inherited functions from std::basic_istream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

Definition at line 427 of file fstream.

4.617.2 Member Typedef Documentation

4.617.2.1 template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type `[inherited]`

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.617.2.2 typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.617.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left

- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.617.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.617.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.617.2.6 typedef _Ios_Seekdir std::ios_base::seekdir `[inherited]`

This is an enumerated type.

`_Ios_Seekdir is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.617.3 Member Enumeration Documentation

4.617.3.1 enum **std::ios_base::event** `[inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.617.4 Constructor & Destructor Documentation

4.617.4.1 template<typename _CharT , typename _Traits > **std::basic_ifstream**< _CharT, _Traits >**::basic_ifstream** ( ) `[inline]`

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 453 of file fstream.

References std::basic_ios< _CharT, _Traits >::init().

4.617.4.2 template<typename _CharT , typename _Traits > **std::basic_ifstream**< _CharT, _Traits >**::basic_ifstream** (const char ∗ *__s*, **ios_base::openmode** *__mode* = **ios_base::in**) `[inline]`, `[explicit]`

Create an input file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

Tip: When using std::string to hold the filename, you must use .c_str() before passing it to this constructor.

Definition at line 467 of file fstream.

References std::basic_ios< _CharT, _Traits >::init(), and std::basic_ifstream< _CharT, _Traits >::open().

4.617.4.3 template<typename _CharT , typename _Traits > **std::basic_ifstream**< _CharT, _Traits >::**basic_ifstream** (const std::string & *__s*, ios_base::openmode *__mode* = ios_base::in) `[inline]`, `[explicit]`

Create an input file stream.

Parameters

__s	std::string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

Definition at line 483 of file fstream.

References std::basic_ios< _CharT, _Traits >::init(), and std::basic_ifstream< _CharT, _Traits >::open().

4.617.4.4 template<typename _CharT , typename _Traits > **std::basic_ifstream**< _CharT, _Traits >::~**basic_ifstream** ( ) `[inline]`

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 498 of file fstream.

4.617.5 Member Function Documentation

4.617.5.1 const **locale**& std::ios_base::_M_getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.617.5.2 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.617.5.3 template<typename _CharT , typename _Traits > void basic_ios< _CharT, _Traits >::clear (iostate *__state* = goodbit) [inherited]

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.617.5.4 template<typename _CharT , typename _Traits > void std::basic_ifstream< _CharT, _Traits >::close ( ) [inline]

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 576 of file fstream.

References std::basic_filebuf< _CharT, _Traits >::close(), std::ios_base::failbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.5 template<typename _CharT , typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt (const basic_ios< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.617.5.6 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.617.5.7 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.617.5.8 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate *__except*) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.617.5.9 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.617.5.10 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.617.5.11 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.617.5.12 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.617.5.13 **fmtflags** std::ios_base::flags (**fmtflags** *__fmtfl*) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.617.5.14 template<typename _CharT, typename _Traits> **streamsize std::basic_istream**< _CharT, _Traits >::gcount ( ) const [inline], [inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.617.5.15 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::int_type **basic_istream**< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.16 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::get (char_type & *__c*) [inherited]

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

∗this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.17 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n, char_type __delim) [inherited]

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.617.5.18 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type * __s, streamsize __n) [inline], [inherited]

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

Definition at line 354 of file istream.

4.617.5.19 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*, char_type *__delim*) `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

4.617.5.20 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*) `[inline]`, `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

∗this

Returns `get`(__sb,widen('\n')).

Definition at line 387 of file istream.

4.617.5.21 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n, char_type __delim) [inherited]

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.617.5.22 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline], [inherited]

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

Definition at line 427 of file istream.

Referenced by std::basic_istream< char >::getline().

4.617.5.23 locale std::ios_base::getloc ( ) const [inline], [inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.617.5.24 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.617.5.25 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize *__n*, int_type *__delim*) [inherited]

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

∗this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.617.5.26 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (**streamsize** *__n*) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.617.5.27 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.28 template<typename _CharT , typename _Traits > **locale basic_ios**< _CharT, _Traits >::imbue (const **locale** & *__loc*) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.617.5.29 template<typename _CharT, typename _Traits> void **basic_ios**< _CharT, _Traits >::init (**basic_streambuf**< _CharT, _Traits > * ___sb_) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.617.5.30 template<typename _CharT , typename _Traits > bool **std::basic_ifstream**< _CharT, _Traits >::is_open ( ) [inline]

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

Definition at line 517 of file fstream.

References std::basic_filebuf< _CharT, _Traits >::is_open().

4.617.5.31 long& std::ios_base::iword (int ___ix_) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.617.5.32 template<typename _CharT, typename _Traits> char **std::basic_ios**< _CharT, _Traits >::narrow (**char_type** ___c_, char ___dfault_) const [inline], [inherited]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 424 of file basic_ios.h.

4.617.5.33 template<typename _CharT , typename _Traits > void **std::basic_ifstream**< _CharT, _Traits >::open (const char * *__s,* **ios_base::openmode** *__mode* = **ios_base::in**) `[inline]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open`(s,__mode|in). If that function fails, `failbit` is set in the stream's error state.

Tip: When using std::string to hold the filename, you must use .c_str() before passing it to this constructor.

Definition at line 538 of file fstream.

References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::failbit, std::ios_base::in, std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::basic_ifstream< _CharT, _Traits >::basic_ifstream().

4.617.5.34 template<typename _CharT , typename _Traits > void **std::basic_ifstream**< _CharT, _Traits >::open (const **std::string** & *__s,* **ios_base::openmode** *__mode* = **ios_base::in**) `[inline]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open`(__s,__mode|in). If that function fails, `failbit` is set in the stream's error state.

Definition at line 558 of file fstream.

References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::failbit, std::ios_base::in, std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.35 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void * ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.617.5.36 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.617.5.37 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 120 of file istream.

4.617.5.38 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 124 of file istream.

4.617.5.39 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 131 of file istream.

4.617.5.40 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & *__n*) `[inline]`, `[inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file istream.

4.617.5.41 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (short & __n) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.42 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

4.617.5.43 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.44 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file istream.

4.617.5.45 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file istream.

4.617.5.46 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file istream.

4.617.5.47 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file istream.

4.617.5.48 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file istream.

4.617.5.49 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file istream.

4.617.5.50 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file istream.

4.617.5.51 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file istream.

4.617.5.52 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& __p) [inline], [inherited]

Basic arithmetic extractors.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file istream.

4.617.5.53 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.54 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::peek (void) [inherited]

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.55 streamsize std::ios_base::precision ( ) const [inline], [inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.617.5.56 streamsize std::ios_base::precision (streamsize __prec) [inline], [inherited]

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.617.5.57 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::putback (char_type *__c*) `[inherited]`

Unextracting a single character.

Parameters

__c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

Referenced by std::operator>>().

4.617.5.58 void∗& std::ios_base::pword (int *__ix*) `[inline]`, `[inherited]`

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.617.5.59 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * __sb) [inherited]

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.617.5.60 template<typename _CharT , typename _Traits > __filebuf_type* std::basic_ifstream< _CharT, _Traits >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

Definition at line 509 of file fstream.

4.617.5.61 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.617.5.62 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::read (char_type * __s, streamsize __n) [inherited]

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.63 template<typename _CharT , typename _Traits > streamsize basic_istream< _CharT, _Traits >::readsome (char_type * __s, streamsize __n) [inherited]

Extraction until the buffer is exhausted, but no more.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.64 void std::ios_base::register_callback (event_callback *__fn,* int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.617.5.65 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::seekg (pos_type *__pos*) `[inherited]`

Changing the current read position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.66 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::seekg (off_type *__off,* ios_base::seekdir *__dir*) `[inherited]`

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.67 fmtflags std::ios_base::setf (fmtflags *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.617.5.68 fmtflags std::ios_base::setf (fmtflags *__fmtfl*, fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.617.5.69 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.617.5.70 template<typename _CharT , typename _Traits > int basic_istream< _CharT, _Traits >::sync (void) [inherited]

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.617.5.71 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) [static], [inherited]

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.617.5.72 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::pos_type **basic_istream**< _CharT, _Traits >::tellg (void) [inherited]

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.617.5.73 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie ( ) const [inline], [inherited]

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.617.5.74 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie (**basic_ostream**< _CharT, _Traits > * *__tiestr*) [inline], [inherited]

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.617.5.75 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::unget (void) [inherited]

Unextracting the previous character.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().

4.617.5.76 void std::ios_base::unsetf (fmtflags __mask) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.617.5.77 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline], [inherited]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.617.5.78 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.617.5.79 streamsize std::ios_base::width (streamsize *__wide*) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.617.5.80 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.617.6 Member Data Documentation

4.617.6.1 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic_istream< char >::gcount(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::unget(), and std::basic_istream< char >::~basic_istream().

4.617.6.2 const fmtflags std::ios_base::adjustfield [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.617.6.3 const openmode std::ios_base::app [static], [inherited]

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.617.6.4 const openmode std::ios_base::ate [static], [inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.617.6.5 const iostate std::ios_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.617.6.6 const fmtflags std::ios_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.617.6.7 const **seekdir** std::ios_base::beg `[static], [inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.617.6.8 const **openmode** std::ios_base::binary `[static], [inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.617.6.9 const **fmtflags** std::ios_base::boolalpha `[static], [inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.617.6.10 const **seekdir** std::ios_base::cur `[static], [inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.617.6.11 const **fmtflags** std::ios_base::dec `[static], [inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.617.6.12 const **seekdir** std::ios_base::end `[static], [inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.617.6.13 const **iostate** std::ios_base::eofbit `[static], [inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.617.6.14 const **iostate** std::ios_base::failbit `[static]`, `[inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.617.6.15 const **fmtflags** std::ios_base::fixed `[static]`, `[inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.617.6.16 const **fmtflags** std::ios_base::floatfield `[static]`, `[inherited]`

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.617.6.17 const **iostate** std::ios_base::goodbit `[static]`, `[inherited]`

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.617.6.18 const fmtflags std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.617.6.19 const openmode std::ios_base::in [static], [inherited]

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.617.6.20 const fmtflags std::ios_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.617.6.21 const fmtflags std::ios_base::left [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.617.6.22 const fmtflags std::ios_base::oct [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.617.6.23 const openmode std::ios_base::out [static], [inherited]

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.617.6.24 const fmtflags std::ios_base::right [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.617.6.25 const fmtflags std::ios_base::scientific `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.617.6.26 const fmtflags std::ios_base::showbase `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.617.6.27 const fmtflags std::ios_base::showpoint `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.617.6.28 const fmtflags std::ios_base::showpos `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.617.6.29 const fmtflags std::ios_base::skipws `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.617.6.30 const openmode std::ios_base::trunc `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.617.6.31 const fmtflags std::ios_base::unitbuf `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.617.6.32 const fmtflags std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

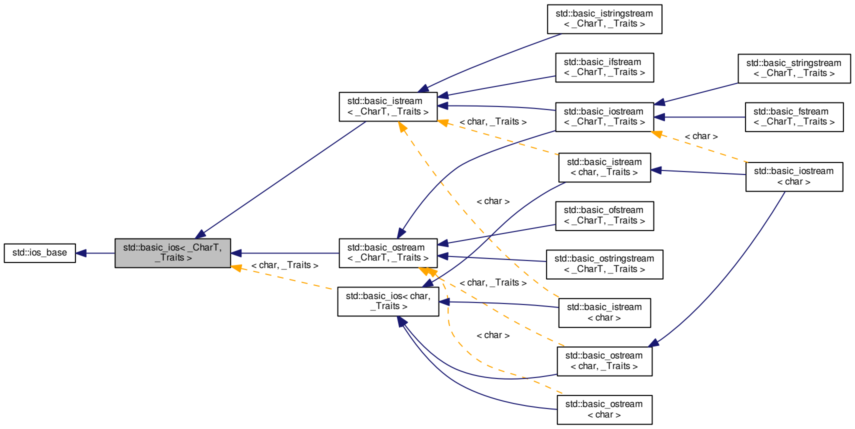
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- fstream

4.618 std::basic_ios< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_ios< _CharT, _Traits >:



Public Types

- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**

- typedef _CharT char_type
- typedef _Traits::int_type int_type
- typedef _Traits::pos_type pos_type
- typedef _Traits::off_type off_type
- typedef _Traits traits_type

- typedef ctype< _CharT > __ctype_type
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ios (basic_streambuf< _CharT, _Traits > ∗__sb)
- virtual ∼basic_ios ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf () const
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- basic_ios ()
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void init (basic_streambuf< _CharT, _Traits > *__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.618.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_ios< _CharT, _Traits >

Template class basic_ios, virtual base class for all stream classes.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 66 of file basic_ios.h.

4.618.2 Member Typedef Documentation

4.618.2.1 template<typename _CharT, typename _Traits> typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type

These are non-standard types.

Definition at line 86 of file basic_ios.h.

4.618.2.2 template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type

These are non-standard types.

Definition at line 90 of file basic_ios.h.

4.618.2.3 template<typename _CharT, typename _Traits> typedef **num_put**<_CharT, **ostreambuf_iterator**<_CharT, _Traits> > **std::basic_ios**< _CharT, _Traits >::**__num_put_type**

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.618.2.4 template<typename _CharT, typename _Traits> typedef _CharT **std::basic_ios**< _CharT, _Traits >::**char_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 75 of file basic_ios.h.

4.618.2.5 typedef void(∗ std::ios_base::event_callback)(**event** __e, **ios_base** &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.618.2.6 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws

- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.618.2.7 template<typename _CharT, typename _Traits> typedef _Traits::int_type **std::basic_ios< _CharT, _Traits >::int_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 76 of file basic_ios.h.

4.618.2.8 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.618.2.9 template<typename _CharT, typename _Traits> typedef _Traits::off_type **std::basic_ios< _CharT, _Traits >::off_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 78 of file basic_ios.h.

4.618.2.10 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.618.2.11 template<typename _CharT, typename _Traits> typedef _Traits::pos_type **std::basic_ios< _CharT, _Traits >::pos_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 77 of file basic_ios.h.

4.618.2.12 typedef _Ios_Seekdir std::ios_base::seekdir `[inherited]`

This is an enumerated type.

`_Ios_Seekdir` `is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.618.2.13 template<typename _CharT, typename _Traits> typedef _Traits **std::basic_ios< _CharT, _Traits >::traits_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 79 of file basic_ios.h.

4.618.3 Member Enumeration Documentation

4.618.3.1 enum **std::ios_base::event** `[inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.618.4 Constructor & Destructor Documentation

4.618.4.1 template<typename _CharT, typename _Traits> **std::basic_ios< _CharT, _Traits >::basic_ios (basic_streambuf< _CharT, _Traits > ∗ *__sb*)** `[inline]`, `[explicit]`

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 264 of file basic_ios.h.

4.618.4.2 template<typename _CharT, typename _Traits> virtual **std::basic_ios< _CharT, _Traits >::∼basic_ios ( )** `[inline]`, `[virtual]`

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

Definition at line 276 of file basic_ios.h.

4.618.4.3 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >**::basic_ios** ( ) `[inline]`, `[protected]`

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 454 of file basic_ios.h.

4.618.5 Member Function Documentation

4.618.5.1 const **locale**& std::ios_base::_M_getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.618.5.2 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const `[inline]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.618.5.3 template<typename _CharT , typename _Traits > void **basic_ios**< _CharT, _Traits >::clear (**iostate** *__state* = **goodbit**)

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.618.5.4 template<typename _CharT , typename _Traits > **basic_ios**< _CharT, _Traits > & **basic_ios**< _CharT, _Traits >::copyfmt (const **basic_ios**< _CharT, _Traits > & *__rhs*)

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.618.5.5 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::eof ( ) const [inline]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.618.5.6 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::exceptions ( ) const [inline]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.618.5.7 template<typename _CharT, typename _Traits> void **std::basic_ios**< _CharT, _Traits >::exceptions (**iostate** *__except*) [inline]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.618.5.8 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::fail ( ) const [inline]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.618.5.9 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill ( ) const [inline]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.618.5.10 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill (char_type *__ch*) [inline]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.618.5.11 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.618.5.12 fmtflags std::ios_base::flags (fmtflags *__fmtfl*) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.618.5.13 locale std::ios_base::getloc ( ) const [inline], [inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.618.5.14 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::good ( ) const [inline]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.618.5.15 template<typename _CharT , typename _Traits > **locale basic_ios**< _CharT, _Traits >::imbue (const **locale** & *__loc*)

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.618.5.16 template<typename _CharT, typename _Traits> void **basic_ios**< _CharT, _Traits >::init (**basic_streambuf**< _CharT, _Traits > * *__sb*) [protected]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.618.5.17 long& std::ios_base::iword (int *__ix*) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.618.5.18 template<typename _CharT, typename _Traits> char **std::basic_ios**< _CharT, _Traits >::narrow (**char_type** *__c*, char *__dfault*) const [inline]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 424 of file basic_ios.h.

4.618.5.19 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void * ( ) const [inline]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.618.5.20 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const [inline]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.618.5.21 streamsize std::ios_base::precision ( ) const [inline], [inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.618.5.22 streamsize std::ios_base::precision (streamsize *__prec*) [inline], [inherited]

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.618.5.23 void∗& std::ios_base::pword (int *__ix*) [inline], [inherited]

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.618.5.24 template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>∗ std::basic_ios< _CharT, _Traits >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.618.5.25 template<typename _CharT, typename _Traits> **basic_streambuf**< _CharT, _Traits > ∗ **basic_ios**< _CharT, _Traits >::rdbuf (**basic_streambuf**< _CharT, _Traits > ∗ *__sb*)

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.618.5.26 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::rdstate ( ) const [inline]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.618.5.27 void std::ios_base::register_callback (**event_callback** *__fn*, int *__index*) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.618.5.28 fmtflags std::ios_base::setf (fmtflags *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.618.5.29 fmtflags std::ios_base::setf (fmtflags *__fmtfl*, fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.618.5.30 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.618.5.31 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.618.5.32 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const `[inline]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.618.5.33 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (basic_ostream< _CharT, _Traits > * *__tiestr*) `[inline]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.618.5.34 void std::ios_base::unsetf (fmtflags __mask) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.618.5.35 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.618.5.36 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.618.5.37 streamsize std::ios_base::width (streamsize *__wide*) `[inline]`, `[inherited]`

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.618.5.38 static int std::ios_base::xalloc ( ) throw () `[static]`, `[inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.618.6 Member Data Documentation

4.618.6.1 const fmtflags std::ios_base::adjustfield `[static]`, `[inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.618.6.2 const openmode std::ios_base::app `[static]`, `[inherited]`

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.618.6.3 const openmode std::ios_base::ate `[static]`, `[inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.618.6.4 const **iostate** std::ios_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.618.6.5 const **fmtflags** std::ios_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.618.6.6 const **seekdir** std::ios_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.618.6.7 const **openmode** std::ios_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.618.6.8 const **fmtflags** std::ios_base::boolalpha [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.618.6.9 const **seekdir** std::ios_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.618.6.10 const fmtflags std::ios_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.618.6.11 const seekdir std::ios_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.618.6.12 const iostate std::ios_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.618.6.13 const iostate std::ios_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.618.6.14 const fmtflags std::ios_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.618.6.15 const fmtflags std::ios_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.618.6.16 const iostate std::ios_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.618.6.17 const fmtflags std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.618.6.18 const openmode std::ios_base::in [static], [inherited]

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.618.6.19 const fmtflags std::ios_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.618.6.20 const fmtflags std::ios_base::left [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.618.6.21 const fmtflags std::ios_base::oct [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.618.6.22 const openmode std::ios_base::out [static], [inherited]

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.618.6.23 const fmtflags std::ios_base::right [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.618.6.24 const fmtflags std::ios_base::scientific [static], [inherited]

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.618.6.25 const fmtflags std::ios_base::showbase [static], [inherited]

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.618.6.26 const fmtflags std::ios_base::showpoint [static], [inherited]

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.618.6.27 const fmtflags std::ios_base::showpos [static], [inherited]

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.618.6.28 const fmtflags std::ios_base::skipws [static], [inherited]

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.618.6.29 const openmode std::ios_base::trunc `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.618.6.30 const fmtflags std::ios_base::unitbuf `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.618.6.31 const fmtflags std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

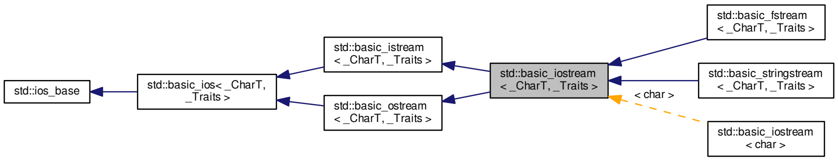
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- basic_ios.h
- basic_ios.tcc

4.619 std::basic_iostream< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_iostream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Traits::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_iostream (basic_streambuf< _CharT, _Traits > ∗__sb)
- virtual ∼basic_iostream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()

- streamsize gcount () const
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (const void ∗__p)
- __ostream_type & operator<< (__streambuf_type ∗__sb)
- __istream_type & operator>> (void ∗&__p)
- __istream_type & operator>> (__streambuf_type ∗__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf () const
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (pos_type)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(∗__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(∗__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(∗__pf)(ios_base &))

Extractors

All the `operator>>` *functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)

  - __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - int_type get ()
  - __istream_type & get (char_type &__c)
  - __istream_type & get (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & get (char_type ∗__s, streamsize __n)
  - __istream_type & get (__streambuf_type &__sb, char_type __delim)
  - __istream_type & get (__streambuf_type &__sb)
  - __istream_type & getline (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & getline (char_type ∗__s, streamsize __n)
  - __istream_type & ignore (streamsize __n, int_type __delim)
  - __istream_type & ignore (streamsize __n)
  - __istream_type & ignore ()
  - int_type peek ()
  - __istream_type & read (char_type ∗__s, streamsize __n)
  - streamsize readsome (char_type ∗__s, streamsize __n)
  - __istream_type & putback (char_type __c)
  - __istream_type & unget ()
  - int sync ()
  - pos_type tellg ()
  - __istream_type & seekg (pos_type)
  - __istream_type & seekg (off_type, ios_base::seekdir)

- operator void ∗ () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(∗__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(∗__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(∗__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __ostream_type & operator<< (long __n)
  - __ostream_type & operator<< (unsigned long __n)

  - __ostream_type & operator<< (bool __n)
  - __ostream_type & operator<< (short __n)
  - __ostream_type & operator<< (unsigned short __n)
  - __ostream_type & operator<< (int __n)
  - __ostream_type & operator<< (unsigned int __n)
  - __ostream_type & operator<< (long long __n)
  - __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

  - __ostream_type & put (char_type __c)
  - void _M_write (const char_type *__s, streamsize __n)
  - __ostream_type & write (const char_type *__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left

- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
  __ostream_type & **_M_insert** (_ValueT __v)
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.619.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_iostream< _CharT, _Traits >

Template class basic_iostream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 795 of file istream.

4.619.2 Member Typedef Documentation

4.619.2.1 template<typename _CharT, typename _Traits> typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type [inherited]

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.619.2.2 typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.619.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]

This is a bitmask type.

`_Ios_Fmtflags` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct

- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.619.2.4 typedef _Ios_Iostate std::ios_base::iostate [inherited]

This is a bitmask type.

`_Ios_Iostate is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.619.2.5 typedef _Ios_Openmode std::ios_base::openmode [inherited]

This is a bitmask type.

`_Ios_Openmode is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.619.2.6 typedef _Ios_Seekdir std::ios_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.619.3 Member Enumeration Documentation

4.619.3.1 enum **std::ios_base::event** [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.619.4 Constructor & Destructor Documentation

4.619.4.1 template<typename _CharT, typename _Traits> **std::basic_iostream**< _CharT, _Traits >::**basic_iostream** (**basic_streambuf**< _CharT, _Traits > ∗ ___sb) [inline], [explicit]

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 820 of file istream.

4.619.4.2 template<typename _CharT, typename _Traits> virtual **std::basic_iostream**< _CharT, _Traits >::∼**basic_iostream** ( ) [inline], [virtual]

Destructor does nothing.

Definition at line 827 of file istream.

4.619.5 Member Function Documentation

4.619.5.1 const **locale**& std::ios_base::_M_getloc ( ) const [inline], [inherited]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.619.5.2 template<typename _CharT, typename _Traits> void **std::basic_ostream**< _CharT, _Traits >::_M_write (const char_type * *__s*, streamsize *__n*) [inline], [inherited]

Core write functionality, without sentry.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Definition at line 311 of file ostream.

Referenced by std::basic_ostream< _CharT, _Traits >::write().

4.619.5.3 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.619.5.4 template<typename _CharT , typename _Traits > void **basic_ios**< _CharT, _Traits >::clear (iostate *__state* = goodbit) [inherited]

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.619.5.5 template<typename _CharT , typename _Traits > **basic_ios**< _CharT, _Traits > & **basic_ios**< _CharT, _Traits >::copyfmt (const **basic_ios**< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.619.5.6 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.619.5.7 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const `[inline]`, `[inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.619.5.8 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate *__except*) `[inline]`, `[inherited]`

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.619.5.9 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::fail ( ) const [inline], [inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.619.5.10 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.619.5.11 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill (**char_type** *__ch*) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.619.5.12 fmtflags std::ios_base::flags ( ) const `[inline]`, `[inherited]`

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.619.5.13 fmtflags std::ios_base::flags (fmtflags *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.619.5.14 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::flush ( ) `[inherited]`

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::flush().

4.619.5.15 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.619.5.16 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.17 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type & __c) [inherited]

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

∗this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.18 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type ∗ __s, streamsize __n, char_type __delim) [inherited]

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.619.5.19 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type ∗ *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

Definition at line 354 of file istream.

4.619.5.20 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*, char_type *__delim*) `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

4.619.5.21 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*) `[inline]`, `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

∗this

Returns `get`(__sb,widen('\n')).

Definition at line 387 of file istream.

4.619.5.22 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline (char_type ∗ *__s*, streamsize *__n*, char_type *__delim*) `[inherited]`

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

∗this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.619.5.23 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type ∗ *__s*, streamsize *__n*) `[inline]`, `[inherited]`

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.

Returns

∗this

Returns `getline`(__s,__n,widen('\n')).

Definition at line 427 of file istream.

Referenced by std::basic_istream< char >::getline().

4.619.5.24 locale std::ios_base::getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.619.5.25 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.619.5.26 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize __n, int_type __delim) [inherited]

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

∗this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.619.5.27 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize __n) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.619.5.28 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.29 template<typename _CharT , typename _Traits > locale basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.619.5.30 template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.619.5.31 long& std::ios_base::iword (int *__ix*) `[inline]`, `[inherited]`

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.619.5.32 template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow (char_type *__c*, char *__dfault*) const `[inline]`, `[inherited]`

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 424 of file basic_ios.h.

4.619.5.33 template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::operator void * ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.619.5.34 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! ( ) const [inline], [inherited]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.619.5.35 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ostream_type &(*)(__ostream_type &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 108 of file ostream.

4.619.5.36 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (__ios_type &(*)(__ios_type &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 117 of file ostream.

4.619.5.37 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 127 of file ostream.

4.619.5.38 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

4.619.5.39 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.619.5.40 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<<(bool *__n*) `[inline]`, `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.619.5.41 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::operator<<(short *__n*) `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.619.5.42 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<<(unsigned short *__n*) `[inline]`, `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.619.5.43 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.619.5.44 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

4.619.5.45 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

4.619.5.46 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

4.619.5.47 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

4.619.5.48 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

4.619.5.49 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.619.5.50 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * __p) [inline], [inherited]

Pointer arithmetic inserters.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.619.5.51 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.52 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) *__pf*) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 120 of file istream.

4.619.5.53 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) *__pf*) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 124 of file istream.

4.619.5.54 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) *__pf*) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 131 of file istream.

4.619.5.55 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & *__n*) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file istream.

4.619.5.56 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (short & *__n*) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.57 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

4.619.5.58 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.59 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file istream.

4.619.5.60 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file istream.

4.619.5.61 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file istream.

4.619.5.62 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file istream.

4.619.5.63 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file istream.

4.619.5.64 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file istream.

4.619.5.65 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file istream.

4.619.5.66 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file istream.

4.619.5.67 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& __p) [inline], [inherited]

Basic arithmetic extractors.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file istream.

4.619.5.68 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.69 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::int_type **basic_istream**< _CharT, _Traits >::peek (void) `[inherited]`

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.70 **streamsize** std::ios_base::precision ( ) const `[inline]`, `[inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.619.5.71 **streamsize** std::ios_base::precision (**streamsize** *__prec*) `[inline]`, `[inherited]`

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.619.5.72 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::put (char_type *__c*) `[inherited]`

Simple insertion.

Parameters

__c	The character to insert.

Returns

∗this

Tries to insert `__c`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.619.5.73 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::putback (char_type __c) `[inherited]`

Unextracting a single character.

Parameters

__c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

Referenced by std::operator>>().

4.619.5.74 void∗& std::ios_base::pword (int __ix) `[inline]`, `[inherited]`

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.619.5.75 template<typename _CharT, typename _Traits> **basic_streambuf**<_CharT, _Traits>∗ **std::basic_ios**< _CharT, _Traits >::rdbuf ( ) const [inline], [inherited]

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.619.5.76 template<typename _CharT, typename _Traits> **basic_streambuf**< _CharT, _Traits > ∗ **basic_ios**< _CharT, _Traits >::rdbuf (**basic_streambuf**< _CharT, _Traits > ∗ *__sb*) [inherited]

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.619.5.77 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.619.5.78 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::read (char_type * __s, **streamsize** __n) [inherited]

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.79 template<typename _CharT , typename _Traits > **streamsize basic_istream**< _CharT, _Traits >::readsome (char_type * __s, **streamsize** __n) [inherited]

Extraction until the buffer is exhausted, but no more.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.80 void std::ios_base::register_callback (**event_callback** *__fn*, int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.619.5.81 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (pos_type *__pos*) `[inherited]`

Changing the current read position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.82 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (off_type *__off*, **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.83 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (pos_type *__pos*) `[inherited]`

Changing the current write position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.84 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (off_type *__off*, **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.85 fmtflags std::ios_base::setf (fmtflags *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.619.5.86 fmtflags std::ios_base::setf (fmtflags *__fmtfl,* fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.619.5.87 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.619.5.88 template<typename _CharT , typename _Traits > int basic_istream< _CharT, _Traits >::sync (void) [inherited]

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.619.5.89 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) [static], [inherited]

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1 html`

4.619.5.90 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::pos_type **basic_istream**< _CharT, _Traits >::tellg (void) `[inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in).`

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.619.5.91 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits >::pos_type **basic_ostream**< _CharT, _Traits >::tellp ( ) `[inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out).`

Definition at line 237 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.619.5.92 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie ( ) const `[inline]`, `[inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.619.5.93 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (basic_ostream< _CharT, _Traits > * *__tiestr*) [inline], [inherited]

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.619.5.94 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::unget (void) [inherited]

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().

4.619.5.95 void std::ios_base::unsetf (fmtflags *__mask*) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.619.5.96 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline], [inherited]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.619.5.97 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.619.5.98 streamsize std::ios_base::width (streamsize __wide) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.619.5.99 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]

Character string insertion.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Returns

*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic_ostream< _CharT, _Traits >::_M_write(), and std::ios_base::badbit.

4.619.5.100 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.619.6 Member Data Documentation

4.619.6.1 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic_istream< char >::gcount(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::unget(), and std::basic_istream< char >::∼basic_istream().

4.619.6.2 const **fmtflags** std::ios_base::adjustfield `[static]`, `[inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.619.6.3 const **openmode** std::ios_base::app `[static]`, `[inherited]`

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.619.6.4 const **openmode** std::ios_base::ate `[static]`, `[inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.619.6.5 const **iostate** std::ios_base::badbit `[static]`, `[inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.619.6.6 const **fmtflags** std::ios_base::basefield `[static]`, `[inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.619.6.7 const **seekdir** std::ios_base::beg `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.619.6.8 const **openmode** std::ios_base::binary `[static], [inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.619.6.9 const **fmtflags** std::ios_base::boolalpha `[static], [inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.619.6.10 const **seekdir** std::ios_base::cur `[static], [inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.619.6.11 const **fmtflags** std::ios_base::dec `[static], [inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.619.6.12 const **seekdir** std::ios_base::end `[static], [inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.619.6.13 const **iostate** std::ios_base::eofbit `[static], [inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.619.6.14 const iostate std::ios_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.619.6.15 const fmtflags std::ios_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.619.6.16 const fmtflags std::ios_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.619.6.17 const iostate std::ios_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.619.6.18 const fmtflags std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.619.6.19 const openmode std::ios_base::in [static], [inherited]

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.619.6.20 const fmtflags std::ios_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.619.6.21 const fmtflags std::ios_base::left [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.619.6.22 const fmtflags std::ios_base::oct [static], [inherited]

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.619.6.23 const openmode std::ios_base::out [static], [inherited]

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.619.6.24 const fmtflags std::ios_base::right [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.619.6.25 const fmtflags std::ios_base::scientific [static], [inherited]

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.619.6.26 const **fmtflags** std::ios_base::showbase `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.619.6.27 const **fmtflags** std::ios_base::showpoint `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.619.6.28 const **fmtflags** std::ios_base::showpos `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.619.6.29 const **fmtflags** std::ios_base::skipws `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.619.6.30 const **openmode** std::ios_base::trunc `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.619.6.31 const **fmtflags** std::ios_base::unitbuf `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.619.6.32 const **fmtflags** std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

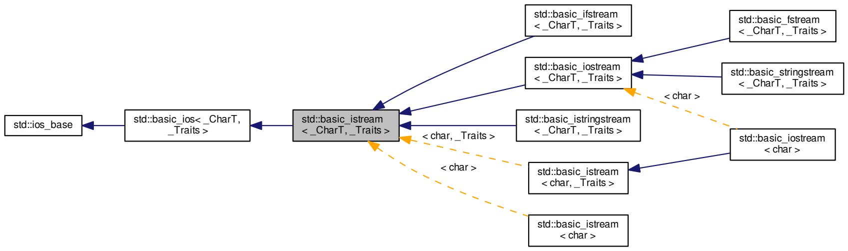
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- istream

4.620 std::basic_istream< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_istream< _CharT, _Traits >:



Classes

- class sentry

    Performs setup work for input streams.

Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Traits::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_istream (__streambuf_type *__sb)
- virtual ~basic_istream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __istream_type & operator>> (void *&__p)
- __istream_type & operator>> (__streambuf_type *__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))

- __istream_type & operator>> (ios_base &(∗__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __istream_type & operator>> (bool &__n)
  - __istream_type & operator>> (short &__n)
  - __istream_type & operator>> (unsigned short &__n)
  - __istream_type & operator>> (int &__n)
  - __istream_type & operator>> (unsigned int &__n)
  - __istream_type & operator>> (long &__n)
  - __istream_type & operator>> (unsigned long &__n)
  - __istream_type & operator>> (long long &__n)
  - __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - int_type get ()
  - __istream_type & get (char_type &__c)
  - __istream_type & get (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & get (char_type ∗__s, streamsize __n)
  - __istream_type & get (__streambuf_type &__sb, char_type __delim)
  - __istream_type & get (__streambuf_type &__sb)
  - __istream_type & getline (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & getline (char_type ∗__s, streamsize __n)
  - __istream_type & ignore (streamsize __n, int_type __delim)
  - __istream_type & ignore (streamsize __n)
  - __istream_type & ignore ()
  - int_type peek ()
  - __istream_type & read (char_type ∗__s, streamsize __n)
  - streamsize readsome (char_type ∗__s, streamsize __n)
  - __istream_type & putback (char_type __c)
  - __istream_type & unget ()
  - int sync ()
  - pos_type tellg ()
  - __istream_type & seekg (pos_type)
  - __istream_type & seekg (off_type, ios_base::seekdir)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void init (basic_streambuf< _CharT, _Traits > *__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**

4.620.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_istream< _CharT, _Traits >

Template class basic_istream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from basic_streambuf to do the actual input.

Definition at line 58 of file istream.

4.620.2 Member Typedef Documentation

4.620.2.1 template<typename _CharT, typename _Traits> typedef **num_put**<_CharT, **ostreambuf_iterator**<_CharT, _Traits> > **std::basic_ios**< _CharT, _Traits >::**__num_put_type** `[inherited]`

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.620.2.2 typedef void(∗ std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.620.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.620.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

_Ios_Iostate `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.620.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

_Ios_Openmode `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.620.2.6 typedef _Ios_Seekdir std::ios_base::seekdir `[inherited]`

This is an enumerated type.

_Ios_Seekdir `is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.620.3 Member Enumeration Documentation

4.620.3.1 enum std::ios_base::event `[inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.620.4 Constructor & Destructor Documentation

4.620.4.1 template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::basic_istream (__streambuf_type * *__sb*) [inline], [explicit]

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 93 of file istream.

4.620.4.2 template<typename _CharT, typename _Traits> virtual std::basic_istream< _CharT, _Traits >::∼basic_istream ( ) [inline], [virtual]

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 103 of file istream.

4.620.5 Member Function Documentation

4.620.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline], [inherited]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.620.5.2 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.620.5.3 template<typename _CharT , typename _Traits > void basic_ios< _CharT, _Traits >::clear (iostate *__state* = goodbit) [inherited]

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.620.5.4 template< typename _CharT , typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt (const basic_ios< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.620.5.5 template< typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.620.5.6 template< typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.620.5.7 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate __except) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.620.5.8 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.620.5.9 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.620.5.10 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.620.5.11 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.620.5.12 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.620.5.13 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.620.5.14 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get (void)

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.15 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type & *__c*)

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

∗this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.16 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type ∗ __s, streamsize __n, char_type __delim)

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.620.5.17 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type ∗ __s, streamsize __n) `[inline]`

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

Definition at line 354 of file istream.

4.620.5.18 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb,* char_type *__delim*)

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

4.620.5.19 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*) `[inline]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

∗this

Returns `get`(__sb,widen('\n')).

Definition at line 387 of file istream.

4.620.5.20 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n, char_type __delim)

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.620.5.21 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * __s, streamsize __n) [inline]

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

Definition at line 427 of file istream.

Referenced by std::basic_istream< char >::getline().

4.620.5.22 **locale** std::ios_base::getloc ( ) const `[inline], [inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.620.5.23 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::good ( ) const `[inline], [inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.620.5.24 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (**streamsize** *__n*, int_type *__delim*)

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.620.5.25 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (**streamsize** *__n*)

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.620.5.26 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (void)

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.27 template<typename _CharT , typename _Traits > **locale basic_ios**< _CharT, _Traits >::imbue (**const locale &** *__loc*) `[inherited]`

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.620.5.28 template<typename _CharT, typename _Traits> void **basic_ios**< _CharT, _Traits >::init (**basic_streambuf**< _CharT, _Traits > * *__sb*) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.620.5.29 long& std::ios_base::iword (int *__ix*) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.620.5.30 template<typename _CharT, typename _Traits> char **std::basic_ios**< _CharT, _Traits >::narrow (**char_type** *__c*, char *__dfault*) const [inline], [inherited]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 424 of file basic_ios.h.

4.620.5.31 template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::operator void ∗ ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.620.5.32 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.620.5.33 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(∗)(__istream_type &) *__pf*) `[inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 120 of file istream.

4.620.5.34 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(∗)(__ios_type &) *__pf*) `[inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 124 of file istream.

4.620.5.35 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(∗)(ios_base &) *__pf*) `[inline]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 131 of file istream.

4.620.5.36 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & *__n*) `[inline]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file istream.

4.620.5.37 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (short & __n)

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.38 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) `[inline]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

4.620.5.39 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (int & __n)

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.40 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file istream.

4.620.5.41 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file istream.

4.620.5.42 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file istream.

4.620.5.43 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file istream.

4.620.5.44 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file istream.

4.620.5.45 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file istream.

4.620.5.46 template<typename _CharT, typename _Traits> **__istream_type& std::basic_istream**< _CharT, _Traits >::operator>> (double & *__f*) [inline]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file istream.

4.620.5.47 template<typename _CharT, typename _Traits> **__istream_type& std::basic_istream**< _CharT, _Traits >::operator>> (long double & *__f*) [inline]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file istream.

4.620.5.48 template<typename _CharT, typename _Traits> **__istream_type& std::basic_istream**< _CharT, _Traits >::operator>> (void *& *__p*) [inline]

Basic arithmetic extractors.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file istream.

4.620.5.49 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::operator>> (**__streambuf_type** * *__sb*)

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.50 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::peek (void)

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.51 streamsize std::ios_base::precision ( ) const `[inline]`, `[inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.620.5.52 streamsize std::ios_base::precision (streamsize *__prec*) `[inline]`, `[inherited]`

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.620.5.53 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::putback (char_type *__c*)

Unextracting a single character.

Parameters

__c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

Referenced by std::operator>>().

4.620.5.54 void∗& std::ios_base::pword (int *__ix*) `[inline]`, `[inherited]`

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.620.5.55 template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::rdbuf ( ) const [inline], [inherited]

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.620.5.56 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > * basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > * *__sb*) [inherited]

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.620.5.57 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.620.5.58 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::read (char_type * __s, streamsize __n)

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.59 template<typename _CharT , typename _Traits > streamsize basic_istream< _CharT, _Traits >::readsome (char_type * __s, streamsize __n)

Extraction until the buffer is exhausted, but no more.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.60 void std::ios_base::register_callback (**event_callback** *__fn,* int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.620.5.61 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (pos_type *__pos*)

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.62 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (off_type *__off,* **ios_base::seekdir** *__dir*)

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.63 **fmtflags** std::ios_base::setf (**fmtflags** *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.620.5.64 **fmtflags** std::ios_base::setf (**fmtflags** *__fmtfl,* **fmtflags** *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.620.5.65 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.620.5.66 template<typename _CharT , typename _Traits > int basic_istream< _CharT, _Traits >::sync (void)

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.620.5.67 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.620.5.68 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg (void)

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.620.5.69 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const `[inline]`, `[inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.620.5.70 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (basic_ostream< _CharT, _Traits > * *__tiestr*) `[inline]`, `[inherited]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.620.5.71 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::unget (void)

Unextracting the previous character.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().

4.620.5.72 void std::ios_base::unsetf (fmtflags *__mask*) `[inline]`, `[inherited]`

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.620.5.73 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char *__c*) const `[inline]`, `[inherited]`

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.620.5.74 **streamsize** std::ios_base::width ( ) const `[inline]`, `[inherited]`

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.620.5.75 **streamsize** std::ios_base::width (**streamsize** *__wide*) `[inline]`, `[inherited]`

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.620.5.76 static int std::ios_base::xalloc ( ) throw () `[static]`, `[inherited]`

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.620.6 Member Data Documentation

4.620.6.1 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount `[protected]`

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic_istream< char >::gcount(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::unget(), and std::basic_istream< char >::∼basic_istream().

4.620.6.2 const fmtflags std::ios_base::adjustfield `[static]`, `[inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.620.6.3 const openmode std::ios_base::app `[static]`, `[inherited]`

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.620.6.4 const openmode std::ios_base::ate `[static]`, `[inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.620.6.5 const iostate std::ios_base::badbit `[static]`, `[inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std-

::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.620.6.6 const **fmtflags** std::ios_base::basefield `[static]`, `[inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.620.6.7 const **seekdir** std::ios_base::beg `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.620.6.8 const **openmode** std::ios_base::binary `[static]`, `[inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.620.6.9 const **fmtflags** std::ios_base::boolalpha `[static]`, `[inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.620.6.10 const **seekdir** std::ios_base::cur `[static]`, `[inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.620.6.11 const **fmtflags** std::ios_base::dec `[static]`, `[inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.620.6.12 const **seekdir** std::ios_base::end `[static]`, `[inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.620.6.13 const **iostate** std::ios_base::eofbit `[static]`, `[inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.620.6.14 const **iostate** std::ios_base::failbit `[static]`, `[inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.620.6.15 const **fmtflags** std::ios_base::fixed `[static]`, `[inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.620.6.16 const **fmtflags** std::ios_base::floatfield `[static]`, `[inherited]`

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.620.6.17 const **iostate** std::ios_base::goodbit `[static]`, `[inherited]`

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits

>::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.620.6.18 const fmtflags std::ios_base::hex `[static], [inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.620.6.19 const openmode std::ios_base::in `[static], [inherited]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.620.6.20 const fmtflags std::ios_base::internal `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.620.6.21 const fmtflags std::ios_base::left `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.620.6.22 const fmtflags std::ios_base::oct `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.620.6.23 const openmode std::ios_base::out `[static], [inherited]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.620.6.24 const fmtflags std::ios_base::right `[static]`, `[inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.620.6.25 const fmtflags std::ios_base::scientific `[static]`, `[inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.620.6.26 const fmtflags std::ios_base::showbase `[static]`, `[inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.620.6.27 const fmtflags std::ios_base::showpoint `[static]`, `[inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.620.6.28 const fmtflags std::ios_base::showpos `[static]`, `[inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.620.6.29 const fmtflags std::ios_base::skipws `[static]`, `[inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.620.6.30 const openmode std::ios_base::trunc `[static]`, `[inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.620.6.31 const fmtflags std::ios_base::unitbuf `[static]`, `[inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.620.6.32 const fmtflags std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- istream
- istream.tcc

4.621 std::basic_istream< _CharT, _Traits >::sentry Class Reference

Public Types

- typedef __istream_type::__ctype_type **__ctype_type**
- typedef _Traits::int_type **__int_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef _Traits traits_type

Public Member Functions

- sentry (basic_istream< _CharT, _Traits > &__is, bool __noskipws=false)
- operator bool () const

4.621.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_istream< _CharT, _Traits >::sentry

Performs setup work for input streams.

Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations,* although only prefix actions are currently required by the standard.

Definition at line 657 of file istream.

4.621.2 Member Typedef Documentation

4.621.2.1 template<typename _CharT, typename _Traits> typedef _Traits std::basic_istream< _CharT, _Traits >::sentry::traits_type

Easy access to dependent types.

Definition at line 664 of file istream.

4.621.3 Constructor & Destructor Documentation

4.621.3.1 template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::sentry (basic_istream< _CharT, _Traits > & *__is*, bool *__noskipws* = `false`) `[explicit]`

The constructor performs all the work.

Parameters

__is	The input stream to guard.
__noskipws	Whether to consume whitespace or not.

If the stream state is good (*__is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.

The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if *__noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::flags(), std::basic_ios< _CharT, _Traits >::good(), std::ios_base::goodbit, std::__ctype_abstract_base< _CharT >::is(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::ios_base::skipws, std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_ios< _CharT, _Traits >::tie().

4.621.4 Member Function Documentation

4.621.4.1 template<typename _CharT, typename _Traits> std::basic_istream< _CharT, _Traits >::sentry::operator bool ( ) const `[inline]`, `[explicit]`

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

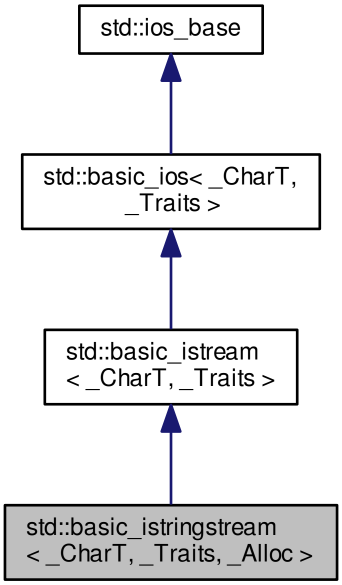
Definition at line 705 of file istream.

The documentation for this class was generated from the following files:

- istream
- istream.tcc

4.622 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference

Inheritance diagram for std::basic_istringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream < char_type, traits_type > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf < _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**

- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_istringstream (ios_base::openmode __mode=ios_base::in)
- basic_istringstream (const __string_type &__str, ios_base::openmode __mode=ios_base::in)
- ∼basic_istringstream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __istream_type & operator>> (void ∗&__p)
- __istream_type & operator>> (__streambuf_type ∗__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- __stringbuf_type ∗ rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)

- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type str () const
- void str (const __string_type &__s)
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(∗__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(∗__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(∗__pf)(ios_base &))

Extractors

All the `operator>>` *functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __istream_type & operator>> (bool &__n)
  - __istream_type & operator>> (short &__n)
  - __istream_type & operator>> (unsigned short &__n)
  - __istream_type & operator>> (int &__n)
  - __istream_type & operator>> (unsigned int &__n)
  - __istream_type & operator>> (long &__n)
  - __istream_type & operator>> (unsigned long &__n)
  - __istream_type & operator>> (long long &__n)
  - __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - int_type get ()
  - __istream_type & get (char_type &__c)
  - __istream_type & get (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & get (char_type ∗__s, streamsize __n)
  - __istream_type & get (__streambuf_type &__sb, char_type __delim)

  - __istream_type & get (__streambuf_type &__sb)
  - __istream_type & getline (char_type ∗__s, streamsize __n, char_type __delim)
  - __istream_type & getline (char_type ∗__s, streamsize __n)
  - __istream_type & ignore (streamsize __n, int_type __delim)
  - __istream_type & ignore (streamsize __n)
  - __istream_type & ignore ()
  - int_type peek ()
  - __istream_type & read (char_type ∗__s, streamsize __n)
  - streamsize readsome (char_type ∗__s, streamsize __n)
  - __istream_type & putback (char_type __c)
  - __istream_type & unget ()
  - int sync ()
  - pos_type tellg ()
  - __istream_type & seekg (pos_type)
  - __istream_type & seekg (off_type, ios_base::seekdir)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint

- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT,
  _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.622.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>class std::basic_istringstream< _CharT, _Traits, _Alloc >

Controlling input for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports reading from objects of type std::basic_string, using the inherited functions from std::basic_istream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

Definition at line 272 of file sstream.

4.622.2 Member Typedef Documentation

4.622.2.1 template<typename _CharT, typename _Traits> typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_put_type `[inherited]`

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.622.2.2 typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.622.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal

- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.622.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.622.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.622.2.6 typedef _Ios_Seekdir std::ios_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.622.3 Member Enumeration Documentation

4.622.3.1 enum **std::ios_base::event** [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.622.4 Constructor & Destructor Documentation

4.622.4.1 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream (ios_base::openmode** *__mode* = **ios_base::in)** [inline], [explicit]

Default constructor starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *__mode*.

Initializes `sb` using __mode|in, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 308 of file sstream.

References std::basic_ios< _CharT, _Traits >::init().

4.622.4.2 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream (** const **__string_type &** *__str,* **ios_base::openmode** *__mode* = **ios_base::in)** [inline], [explicit]

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.

Initializes `sb` using *str* and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 326 of file sstream.

References std::basic_ios< _CharT, _Traits >::init().

4.622.4.3 template<typename _CharT, typename _Traits, typename _Alloc> std::basic_istringstream< _CharT, _Traits, _Alloc >::∼basic_istringstream ( ) `[inline]`

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 337 of file sstream.

4.622.5 Member Function Documentation

4.622.5.1 const locale& std::ios_base::_M_getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.622.5.2 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.622.5.3 template<typename _CharT , typename _Traits > void basic_ios< _CharT, _Traits >::clear (iostate *__state* = goodbit) `[inherited]`

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.622.5.4 template<typename _CharT , typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt (const basic_ios< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.622.5.5 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.622.5.6 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.622.5.7 template<typename _CharT, typename _Traits> void **std::basic_ios**< _CharT, _Traits >::exceptions (**iostate** *__except*) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.622.5.8 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::fail ( ) const [inline], [inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void *(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.622.5.9 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.622.5.10 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.622.5.11 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.622.5.12 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.622.5.13 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.622.5.14 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::int_type **basic_istream**< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.15 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::get (char_type & *__c*) [inherited]

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

∗this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.16 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::get (char_type ∗ *__s*, **streamsize** *__n*, char_type *__delim*) [inherited]

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.622.5.17 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type ∗ *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

Definition at line 354 of file istream.

4.622.5.18 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*, char_type *__delim*) `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

4.622.5.19 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & __sb) [inline], [inherited]

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

∗this

Returns `get`(__sb,widen('\n')).

Definition at line 387 of file istream.

4.622.5.20 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline (char_type ∗ __s, streamsize __n, char_type __delim) [inherited]

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

∗this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.622.5.21 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type * *__s*, streamsize *__n*) `[inline]`, `[inherited]`

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.

Returns

∗this

Returns `getline`(__s,__n,widen('\n')).

Definition at line 427 of file istream.

Referenced by std::basic_istream< char >::getline().

4.622.5.22 locale std::ios_base::getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.622.5.23 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.622.5.24 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize __n, int_type __delim) [inherited]

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

∗this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.622.5.25 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::ignore (streamsize __n) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.622.5.26 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.27 template<typename _CharT , typename _Traits > **locale basic_ios**< _CharT, _Traits >::imbue (const **locale** & *__loc*) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.622.5.28 template<typename _CharT, typename _Traits> void **basic_ios**< _CharT, _Traits >::init (**basic_streambuf**< _CharT, _Traits > * *__sb*) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.622.5.29 long& std::ios_base::iword (int *__ix*) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.622.5.30 template<typename _CharT, typename _Traits> char **std::basic_ios**< _CharT, _Traits >::narrow (**char_type** *__c*, char *__dfault*) const [inline], [inherited]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 424 of file basic_ios.h.

4.622.5.31 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void ∗ ( ) const [inline], [inherited]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.622.5.32 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const [inline], [inherited]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.622.5.33 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) *__pf*) `[inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 120 of file istream.

4.622.5.34 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) *__pf*) `[inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 124 of file istream.

4.622.5.35 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) *__pf*) `[inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 131 of file istream.

4.622.5.36 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & *__n*) `[inline], [inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file istream.

4.622.5.37 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (short & *__n*) `[inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.38 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

4.622.5.39 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.40 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file istream.

4.622.5.41 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file istream.

4.622.5.42 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file istream.

4.622.5.43 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file istream.

4.622.5.44 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file istream.

4.622.5.45 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file istream.

4.622.5.46 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file istream.

4.622.5.47 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file istream.

4.622.5.48 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& __p) [inline], [inherited]

Basic arithmetic extractors.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file istream.

4.622.5.49 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.50 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::int_type **basic_istream**< _CharT, _Traits >::peek (void) [inherited]

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.51 **streamsize** std::ios_base::precision ( ) const [inline], [inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.622.5.52 **streamsize** std::ios_base::precision (**streamsize** *__prec*) [inline], [inherited]

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.622.5.53 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::putback (char_type *__c*) [inherited]

Unextracting a single character.

Parameters

__c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

Referenced by std::operator>>().

4.622.5.54 void∗& std::ios_base::pword (int *__ix*) `[inline]`, `[inherited]`

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.622.5.55 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > ∗ basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > ∗ *__sb*) `[inherited]`

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.622.5.56 template<typename _CharT, typename _Traits, typename _Alloc> __stringbuf_type* **std::basic_istringstream**< _CharT, _Traits, _Alloc >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

Definition at line 348 of file sstream.

4.622.5.57 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.622.5.58 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::read (char_type * *__s*, **streamsize** *__n*) [inherited]

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.59 template<typename _CharT , typename _Traits > streamsize basic_istream< _CharT, _Traits >::readsome (char_type * __s, streamsize __n) [inherited]

Extraction until the buffer is exhausted, but no more.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.60 void std::ios_base::register_callback (event_callback __fn, int __index) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.622.5.61 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::seekg (pos_type __pos) [inherited]

Changing the current read position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.62 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (off_type *__off,* **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.63 **fmtflags** std::ios_base::setf (**fmtflags** *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.622.5.64 fmtflags std::ios_base::setf (fmtflags *__fmtfl*, fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.622.5.65 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.622.5.66 template<typename _CharT, typename _Traits, typename _Alloc> __string_type std::basic_istringstream< _CharT, _Traits, _Alloc >::str ( ) const [inline]

Copying out the string buffer.

Returns

`rdbuf()->str()`

Definition at line 356 of file sstream.

4.622.5.67 template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_istringstream< _CharT, _Traits, _Alloc >::str (const __string_type & __s) [inline]

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

Definition at line 366 of file sstream.

4.622.5.68 template<typename _CharT , typename _Traits > int basic_istream< _CharT, _Traits >::sync (void) [inherited]

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.622.5.69 static bool std::ios_base::sync_with_stdio (bool __sync = `true`) [static], [inherited]

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1
html

4.622.5.70 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits >::pos_type **basic_istream**< _CharT, _Traits >::tellg (void) `[inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.622.5.71 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie ( ) const `[inline]`, `[inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.622.5.72 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie (**basic_ostream**< _CharT, _Traits > * *__tiestr*) `[inline]`, `[inherited]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.622.5.73 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::unget (void) [inherited]

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().

4.622.5.74 void std::ios_base::unsetf (**fmtflags** *__mask*) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.622.5.75 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::widen (char *__c*) const [inline], [inherited]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.622.5.76 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.622.5.77 streamsize std::ios_base::width (streamsize __wide) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.622.5.78 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.622.6 Member Data Documentation

4.622.6.1 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount `[protected]`, `[inherited]`

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic_istream< char >::gcount(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::unget(), and std::basic_istream< char >::∼basic_istream().

4.622.6.2 const fmtflags std::ios_base::adjustfield `[static]`, `[inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.622.6.3 const openmode std::ios_base::app `[static]`, `[inherited]`

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.622.6.4 const openmode std::ios_base::ate `[static]`, `[inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.622.6.5 const iostate std::ios_base::badbit `[static]`, `[inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.622.6.6 const fmtflags std::ios_base::basefield `[static]`, `[inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.622.6.7 const **seekdir** std::ios_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.622.6.8 const **openmode** std::ios_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.622.6.9 const **fmtflags** std::ios_base::boolalpha [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.622.6.10 const **seekdir** std::ios_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.622.6.11 const **fmtflags** std::ios_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.622.6.12 const **seekdir** std::ios_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.622.6.13 const **iostate** std::ios_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.622.6.14 const iostate std::ios_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.622.6.15 const fmtflags std::ios_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.622.6.16 const fmtflags std::ios_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.622.6.17 const iostate std::ios_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.622.6.18 const fmtflags std::ios_base::hex `[static], [inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.622.6.19 const openmode std::ios_base::in `[static], [inherited]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.622.6.20 const fmtflags std::ios_base::internal `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.622.6.21 const fmtflags std::ios_base::left `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.622.6.22 const fmtflags std::ios_base::oct `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.622.6.23 const openmode std::ios_base::out `[static], [inherited]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.622.6.24 const fmtflags std::ios_base::right `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.622.6.25 const fmtflags std::ios_base::scientific `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.622.6.26 const fmtflags std::ios_base::showbase `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.622.6.27 const fmtflags std::ios_base::showpoint `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.622.6.28 const fmtflags std::ios_base::showpos `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.622.6.29 const fmtflags std::ios_base::skipws `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.622.6.30 const openmode std::ios_base::trunc `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.622.6.31 const fmtflags std::ios_base::unitbuf `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.622.6.32 const fmtflags std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

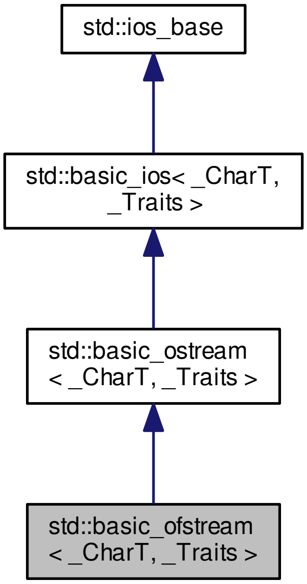
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- sstream

4.623 std::basic_ofstream< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_ofstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf < char_type, traits_type > **__filebuf_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream < char_type, traits_type > **__ostream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)

- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ofstream ()
- basic_ofstream (const char ∗__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)
- basic_ofstream (const std::string &__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)
- ∼basic_ofstream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- void open (const char ∗__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::out|ios_base::trunc)
- __ostream_type & operator<< (const void ∗__p)
- __ostream_type & operator<< (__streambuf_type ∗__sb)

- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- __filebuf_type ∗ rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (pos_type)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(∗__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(∗__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(∗__pf)(ios_base &))

Inserters

All the `operator<<` *functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __ostream_type & operator<< (long __n)
  - __ostream_type & operator<< (unsigned long __n)
  - __ostream_type & operator<< (bool __n)
  - __ostream_type & operator<< (short __n)
  - __ostream_type & operator<< (unsigned short __n)
  - __ostream_type & operator<< (int __n)
  - __ostream_type & operator<< (unsigned int __n)
  - __ostream_type & operator<< (long long __n)
  - __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

  - __ostream_type & put (char_type __c)
  - void _M_write (const char_type ∗__s, streamsize __n)
  - __ostream_type & write (const char_type ∗__s, streamsize __n)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
  __ostream_type & **_M_insert** (_ValueT __v)
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.623.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_ofstream< _CharT, _Traits >

Controlling output for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from named files, using the inherited functions from std::basic_ostream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

Definition at line 599 of file fstream.

4.623.2 Member Typedef Documentation

4.623.2.1 template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type [inherited]

These are non-standard types.

Definition at line 90 of file basic_ios.h.

4.623.2.2 typedef void(∗ std::ios_base::event_callback)(event __e, ios_base &__b, int __i) [inherited]

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.623.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.623.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

_Ios_Iostate `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.623.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

_Ios_Openmode `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.623.2.6 typedef _Ios_Seekdir std::ios_base::seekdir `[inherited]`

This is an enumerated type.

_Ios_Seekdir `is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.623.3 Member Enumeration Documentation

4.623.3.1 enum std::ios_base::event `[inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.623.4 Constructor & Destructor Documentation

4.623.4.1 template<typename _CharT , typename _Traits > **std::basic_ofstream< _CharT, _Traits >::basic_ofstream ( )** [inline]

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 625 of file fstream.

References std::basic_ios< _CharT, _Traits >::init().

4.623.4.2 template<typename _CharT , typename _Traits > **std::basic_ofstream< _CharT, _Traits >::basic_ofstream (** const char * *__s*, **ios_base::openmode** *__mode* = **ios_base::out|ios_base::trunc)** [inline], [explicit]

Create an output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out | ios_base::trunc` is automatically included in *__mode*.

Tip: When using std::string to hold the filename, you must use .c_str() before passing it to this constructor.

Definition at line 640 of file fstream.

References std::basic_ios< _CharT, _Traits >::init(), and std::basic_ofstream< _CharT, _Traits >::open().

4.623.4.3 template<typename _CharT , typename _Traits > **std::basic_ofstream< _CharT, _Traits >::basic_ofstream (** const **std::string &** *__s*, **ios_base::openmode** *__mode* = **ios_base::out|ios_base::trunc)** [inline], [explicit]

Create an output file stream.

Parameters

__s	std::string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out | ios_base::trunc` is automatically included in *__mode*.

Definition at line 658 of file fstream.

References std::basic_ios< _CharT, _Traits >::init(), and std::basic_ofstream< _CharT, _Traits >::open().

4.623.4.4 template<typename _CharT , typename _Traits > **std::basic_ofstream< _CharT, _Traits >::∼basic_ofstream ( )** [inline]

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 673 of file fstream.

4.623.5 Member Function Documentation

4.623.5.1 const **locale**& std::ios_base::_M_getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.623.5.2 template<typename _CharT, typename _Traits> void **std::basic_ostream**< _CharT, _Traits >::_M_write (const char_type * *__s*, **streamsize** *__n*) `[inline]`, `[inherited]`

Core write functionality, without sentry.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Definition at line 311 of file ostream.

Referenced by std::basic_ostream< _CharT, _Traits >::write().

4.623.5.3 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.623.5.4 template<typename _CharT , typename _Traits > void **basic_ios**< _CharT, _Traits >::clear (**iostate** *__state* = **goodbit**) `[inherited]`

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.623.5.5 template<typename _CharT , typename _Traits > void **std::basic_ofstream**< _CharT, _Traits >::close ( ) `[inline]`

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 753 of file fstream.

References std::basic_filebuf< _CharT, _Traits >::close(), std::ios_base::failbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.623.5.6 template<typename _CharT , typename _Traits > **basic_ios**< _CharT, _Traits > & **basic_ios**< _CharT, _Traits >::copyfmt (const **basic_ios**< _CharT, _Traits > & *__rhs*) `[inherited]`

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.623.5.7 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::eof ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.623.5.8 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::exceptions ( ) const `[inline]`, `[inherited]`

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.623.5.9 template<typename _CharT, typename _Traits> void **std::basic_ios**< _CharT, _Traits >::exceptions (**iostate** *__except*) `[inline]`, `[inherited]`

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.623.5.10 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::fail ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.623.5.11 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.623.5.12 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type *__ch*) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.623.5.13 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.623.5.14 fmtflags std::ios_base::flags (fmtflags *__fmtfl*) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.623.5.15 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::flush ( ) [inherited]

Synchronizing the stream buffer.

Returns

∗this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::flush().

4.623.5.16 locale std::ios_base::getloc ( ) const [inline], [inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.623.5.17 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.623.5.18 template<typename _CharT , typename _Traits > locale basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.623.5.19 template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.623.5.20 template<typename _CharT , typename _Traits > bool std::basic_ofstream< _CharT, _Traits >::is_open ( ) [inline]

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

Definition at line 692 of file fstream.

References std::basic_filebuf< _CharT, _Traits >::is_open().

4.623.5.21 long& std::ios_base::iword (int __ix) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.623.5.22 template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow (char_type *__c*, char *__dfault*) const [inline], [inherited]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 424 of file basic_ios.h.

4.623.5.23 template<typename _CharT , typename _Traits > void std::basic_ofstream< _CharT, _Traits >::open (const char * *__s*, ios_base::openmode *__mode* = ios_base::out | ios_base::trunc) [inline]

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open`(__s,__mode|out|trunc). If that function fails, `failbit` is set in the stream's error state.

Tip: When using std::string to hold the filename, you must use .c_str() before passing it to this constructor.

Definition at line 713 of file fstream.

References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::failbit, std::basic_filebuf< _CharT, _Traits >::open(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::basic_ofstream< _CharT, _Traits >::basic_ofstream().

4.623.5.24 template<typename _CharT , typename _Traits > void **std::basic_ofstream**< _CharT, _Traits >::open (const std::string & *__s*, ios_base::openmode *__mode* = ios_base::out | ios_base::trunc) `[inline]`

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open`(s,mode|out|trunc). If that function fails, `failbit` is set in the stream's error state.

Definition at line 734 of file fstream.

References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::failbit, std::basic_filebuf< _CharT, _Traits >::open(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::setstate().

4.623.5.25 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void ∗ ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.623.5.26 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.623.5.27 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (__ostream_type &(∗)(__ostream_type &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 108 of file ostream.

4.623.5.28 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (__ios_type &(∗)(__ios_type &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 117 of file ostream.

4.623.5.29 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (ios_base &(∗)(ios_base &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 127 of file ostream.

4.623.5.30 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

4.623.5.31 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.623.5.32 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.623.5.33 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (short __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.623.5.34 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.623.5.35 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.623.5.36 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

4.623.5.37 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

4.623.5.38 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

4.623.5.39 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

4.623.5.40 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

4.623.5.41 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.623.5.42 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * __p) [inline], [inherited]

Pointer arithmetic inserters.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.623.5.43 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) [inherited]

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.623.5.44 streamsize std::ios_base::precision ( ) const [inline], [inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.623.5.45 **streamsize** std::ios_base::precision (**streamsize** *__prec*) [inline], [inherited]

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.623.5.46 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::put (char_type *__c*) [inherited]

Simple insertion.

Parameters

__c	The character to insert.

Returns

∗this

Tries to insert __`c`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.623.5.47 void∗& std::ios_base::pword (int *__ix*) [inline], [inherited]

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.623.5.48 template<typename _CharT, typename _Traits> **basic_streambuf**< _CharT, _Traits > ∗ **basic_ios**< _CharT, _Traits >::rdbuf (**basic_streambuf**< _CharT, _Traits > ∗ *__sb*) [inherited]

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.623.5.49 template<typename _CharT , typename _Traits > **__filebuf_type**∗ **std::basic_ofstream**< _CharT, _Traits >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

Definition at line 684 of file fstream.

4.623.5.50 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.623.5.51 void std::ios_base::register_callback (**event_callback** *__fn,* int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.623.5.52 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (pos_type *__pos*) `[inherited]`

Changing the current write position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.623.5.53 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (off_type *__off,* **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.623.5.54 **fmtflags** std::ios_base::setf (**fmtflags** *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.623.5.55 fmtflags std::ios_base::setf (fmtflags *__fmtfl,* fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.623.5.56 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.623.5.57 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.623.5.58 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp ( ) `[inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.

Definition at line 237 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.623.5.59 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const `[inline]`, `[inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.623.5.60 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (basic_ostream< _CharT, _Traits > * *__tiestr*) `[inline]`, `[inherited]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.623.5.61 void std::ios_base::unsetf (fmtflags __mask) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.623.5.62 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char __c) const [inline], [inherited]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.623.5.63 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.623.5.64 **streamsize** std::ios_base::width (**streamsize** *__wide*) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.623.5.65 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::write (const char_type * *__s*, **streamsize** *__n*) [inherited]

Character string insertion.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Returns

*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic_ostream< _CharT, _Traits >::_M_write(), and std::ios_base::badbit.

4.623.5.66 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.623.6 Member Data Documentation

4.623.6.1 const fmtflags std::ios_base::adjustfield `[static], [inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.623.6.2 const openmode std::ios_base::app `[static], [inherited]`

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.623.6.3 const openmode std::ios_base::ate `[static], [inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.623.6.4 const iostate std::ios_base::badbit `[static], [inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.623.6.5 const fmtflags std::ios_base::basefield `[static], [inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.623.6.6 const **seekdir** std::ios_base::beg `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.623.6.7 const **openmode** std::ios_base::binary `[static]`, `[inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.623.6.8 const **fmtflags** std::ios_base::boolalpha `[static]`, `[inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.623.6.9 const **seekdir** std::ios_base::cur `[static]`, `[inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.623.6.10 const **fmtflags** std::ios_base::dec `[static]`, `[inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.623.6.11 const **seekdir** std::ios_base::end `[static]`, `[inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.623.6.12 const **iostate** std::ios_base::eofbit `[static]`, `[inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits

>::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.623.6.13 const iostate std::ios_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.623.6.14 const fmtflags std::ios_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.623.6.15 const fmtflags std::ios_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.623.6.16 const iostate std::ios_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.623.6.17 const fmtflags std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.623.6.18 const openmode std::ios_base::in `[static], [inherited]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.623.6.19 const fmtflags std::ios_base::internal `[static], [inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.623.6.20 const fmtflags std::ios_base::left `[static], [inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.623.6.21 const fmtflags std::ios_base::oct `[static], [inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.623.6.22 const openmode std::ios_base::out `[static], [inherited]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.623.6.23 const fmtflags std::ios_base::right `[static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.623.6.24 const fmtflags std::ios_base::scientific `[static]`, `[inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.623.6.25 const fmtflags std::ios_base::showbase `[static]`, `[inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.623.6.26 const fmtflags std::ios_base::showpoint `[static]`, `[inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.623.6.27 const fmtflags std::ios_base::showpos `[static]`, `[inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.623.6.28 const fmtflags std::ios_base::skipws `[static]`, `[inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.623.6.29 const openmode std::ios_base::trunc `[static]`, `[inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.623.6.30 const fmtflags std::ios_base::unitbuf `[static]`, `[inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.623.6.31 const fmtflags std::ios_base::uppercase `[static]`, `[inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

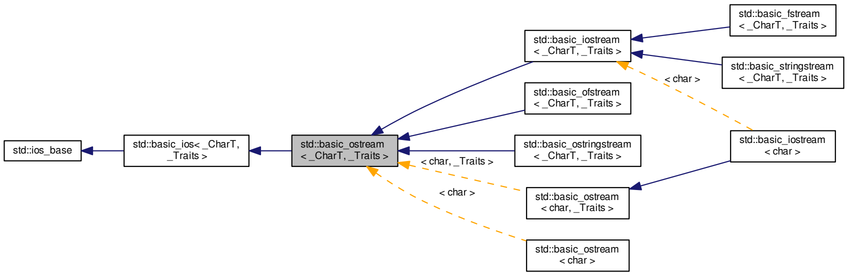
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- fstream

4.624 std::basic_ostream< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_ostream< _CharT, _Traits >:



Classes

- class sentry

  Performs setup work for output streams.

Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Traits::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ostream (__streambuf_type ∗__sb)
- virtual ∼basic_ostream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (const void ∗__p)
- __ostream_type & operator<< (__streambuf_type ∗__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT, _Traits > ∗ rdbuf () const
- basic_streambuf< _CharT, _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (pos_type)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(∗__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(∗__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(∗__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- void _M_write (const char_type ∗__s, streamsize __n)
- __ostream_type & write (const char_type ∗__s, streamsize __n)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
  __ostream_type & **_M_insert** (_ValueT __v)
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**

4.624.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_ostream< _CharT, _Traits >

Template class basic_ostream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from basic_streambuf to do the actual output.

Definition at line 58 of file ostream.

4.624.2 Member Typedef Documentation

4.624.2.1 template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type `[inherited]`

These are non-standard types.

Definition at line 90 of file basic_ios.h.

4.624.2.2 typedef void(∗ std::ios_base::event_callback)(event __e, ios_base &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.624.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.624.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.624.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.624.2.6 typedef _Ios_Seekdir std::ios_base::seekdir `[inherited]`

This is an enumerated type.

`_Ios_Seekdir` `is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.624.3 Member Enumeration Documentation

4.624.3.1 enum std::ios_base::event `[inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.624.4 Constructor & Destructor Documentation

4.624.4.1 template<typename _CharT, typename _Traits> **std::basic_ostream**< _CharT, _Traits >::**basic_ostream** (**__streambuf_type** * *__sb*) [inline], [explicit]

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 84 of file ostream.

4.624.4.2 template<typename _CharT, typename _Traits> virtual **std::basic_ostream**< _CharT, _Traits >::~**basic_ostream** ( ) [inline], [virtual]

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 93 of file ostream.

4.624.5 Member Function Documentation

4.624.5.1 const **locale**& std::ios_base::_M_getloc ( ) const [inline], [inherited]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.624.5.2 template<typename _CharT, typename _Traits> void **std::basic_ostream**< _CharT, _Traits >::_M_write (const char_type * *__s*, **streamsize** *__n*) [inline]

Core write functionality, without sentry.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Definition at line 311 of file ostream.

Referenced by std::basic_ostream< _CharT, _Traits >::write().

4.624.5.3 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.624.5.4 template<typename _CharT , typename _Traits > void basic_ios< _CharT, _Traits >::clear (iostate *__state* = goodbit) [inherited]

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.624.5.5 template<typename _CharT , typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt (const basic_ios< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.624.5.6 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.624.5.7 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.624.5.8 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate *__except*) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.624.5.9 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.624.5.10 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.624.5.11 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.624.5.12 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.624.5.13 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.624.5.14 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::flush ( )

Synchronizing the stream buffer.

Returns

∗this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::flush().

4.624.5.15 locale std::ios_base::getloc ( ) const [inline], [inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.624.5.16 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const [inline], [inherited]

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.624.5.17 template<typename _CharT , typename _Traits > locale basic_ios< _CharT, _Traits >::imbue (const locale & *__loc*) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.624.5.18 template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT, _Traits > * *__sb*) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.624.5.19 long& std::ios_base::iword (int *__ix*) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.624.5.20 template<typename _CharT, typename _Traits> char **std::basic_ios**< _CharT, _Traits >::narrow (**char_type** *__c*, char *__dfault*) const `[inline]`, `[inherited]`

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 424 of file basic_ios.h.

4.624.5.21 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void $*$ ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.624.5.22 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.624.5.23 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ostream_type** &($*$)(**__ostream_type** &) *__pf*) `[inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 108 of file ostream.

4.624.5.24 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ios_type** &($*$)(**__ios_type** &) *__pf*) `[inline]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 117 of file ostream.

4.624.5.25 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) *__pf*) [inline]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 127 of file ostream.

4.624.5.26 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long *__n*) [inline]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

4.624.5.27 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long *__n*) [inline]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.624.5.28 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool *__n*) [inline]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.624.5.29 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (short __n)

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.624.5.30 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) `[inline]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.624.5.31 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (int __n)

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.624.5.32 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

4.624.5.33 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

4.624.5.34 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

4.624.5.35 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

4.624.5.36 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<(float *__f*) `[inline]`

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

4.624.5.37 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<(long double *__f*) `[inline]`

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.624.5.38 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<<(const void * *__p*) `[inline]`

Pointer arithmetic inserters.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.624.5.39 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb)

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.624.5.40 streamsize std::ios_base::precision ( ) const `[inline]`, `[inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.624.5.41 streamsize std::ios_base::precision (streamsize __prec) `[inline]`, `[inherited]`

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.624.5.42 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::put (char_type __c)

Simple insertion.

Parameters

__c	The character to insert.

Returns

∗this

Tries to insert `__c`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.624.5.43 void∗& std::ios_base::pword (int __ix) [inline], [inherited]

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.624.5.44 template<typename _CharT, typename _Traits> basic_streambuf<_CharT, _Traits>∗ std::basic_ios< _CharT, _Traits >::rdbuf ( ) const [inline], [inherited]

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 315 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.624.5.45 template<typename _CharT, typename _Traits> **basic_streambuf**< _CharT, _Traits > ∗ **basic_ios**< _CharT, _Traits >::rdbuf (**basic_streambuf**< _CharT, _Traits > ∗ *__sb*) `[inherited]`

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.624.5.46 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const `[inline]`, `[inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.624.5.47 void std::ios_base::register_callback (**event_callback** *__fn,* int *__index*) [inherited]

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.624.5.48 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (pos_type *__pos*)

Changing the current write position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.624.5.49 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (off_type *__off,* **ios_base::seekdir** *__dir*)

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.624.5.50 **fmtflags** std::ios_base::setf (**fmtflags** *__fmtfl*) [inline], [inherited]

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.624.5.51 fmtflags std::ios_base::setf (fmtflags *__fmtfl,* fmtflags *__mask*) [inline], [inherited]

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.624.5.52 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) [inline], [inherited]

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.624.5.53 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.624.5.54 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits >::pos_type **basic_ostream**< _CharT, _Traits >::tellp ( )

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.

Definition at line 237 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.624.5.55 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie ( ) const `[inline]`, `[inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.624.5.56 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie (**basic_ostream**< _CharT, _Traits > * *__tiestr*) `[inline]`, `[inherited]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.624.5.57 void std::ios_base::unsetf (fmtflags *__mask*) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.624.5.58 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char *__c*) const [inline], [inherited]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.624.5.59 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.624.5.60 **streamsize** std::ios_base::width (**streamsize** *__wide*) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.624.5.61 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::write (const char_type * *__s*, **streamsize** *__n*)

Character string insertion.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Returns

*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic_ostream< _CharT, _Traits >::_M_write(), and std::ios_base::badbit.

4.624.5.62 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.624.6 Member Data Documentation

4.624.6.1 const fmtflags std::ios_base::adjustfield `[static]`, `[inherited]`

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.624.6.2 const openmode std::ios_base::app `[static]`, `[inherited]`

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.624.6.3 const openmode std::ios_base::ate `[static]`, `[inherited]`

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.624.6.4 const iostate std::ios_base::badbit `[static]`, `[inherited]`

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.624.6.5 const fmtflags std::ios_base::basefield `[static]`, `[inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.624.6.6 const **seekdir** std::ios_base::beg `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.624.6.7 const **openmode** std::ios_base::binary `[static]`, `[inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.624.6.8 const **fmtflags** std::ios_base::boolalpha `[static]`, `[inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.624.6.9 const **seekdir** std::ios_base::cur `[static]`, `[inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.624.6.10 const **fmtflags** std::ios_base::dec `[static]`, `[inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.624.6.11 const **seekdir** std::ios_base::end `[static]`, `[inherited]`

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.624.6.12 const **iostate** std::ios_base::eofbit `[static]`, `[inherited]`

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits

>::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.624.6.13 const **iostate** std::ios_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.624.6.14 const **fmtflags** std::ios_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.624.6.15 const **fmtflags** std::ios_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.624.6.16 const **iostate** std::ios_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.624.6.17 const **fmtflags** std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.624.6.18 const **openmode** std::ios_base::in `[static]`, `[inherited]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.624.6.19 const **fmtflags** std::ios_base::internal `[static]`, `[inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.624.6.20 const **fmtflags** std::ios_base::left `[static]`, `[inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.624.6.21 const **fmtflags** std::ios_base::oct `[static]`, `[inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.624.6.22 const **openmode** std::ios_base::out `[static]`, `[inherited]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.624.6.23 const **fmtflags** std::ios_base::right `[static]`, `[inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.624.6.24 const fmtflags std::ios_base::scientific `[static]`, `[inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.624.6.25 const fmtflags std::ios_base::showbase `[static]`, `[inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.624.6.26 const fmtflags std::ios_base::showpoint `[static]`, `[inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.624.6.27 const fmtflags std::ios_base::showpos `[static]`, `[inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.624.6.28 const fmtflags std::ios_base::skipws `[static]`, `[inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.624.6.29 const openmode std::ios_base::trunc `[static]`, `[inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.624.6.30 const fmtflags std::ios_base::unitbuf `[static]`, `[inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.624.6.31 const fmtflags std::ios_base::uppercase `[static]`, `[inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- ostream

- ostream.tcc

4.625 std::basic_ostream< _CharT, _Traits >::sentry Class Reference

Public Member Functions

- sentry (basic_ostream< _CharT, _Traits > &__os)
- ∼sentry ()
- operator bool () const

4.625.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_ostream< _CharT, _Traits >::sentry

Performs setup work for output streams.

Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 400 of file ostream.

4.625.2 Constructor & Destructor Documentation

4.625.2.1 template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::sentry::sentry (basic_ostream< _CharT, _Traits > & __os) [explicit]

The constructor performs preparatory work.

Parameters

__os	The output stream to guard.

If the stream state is good (*__os.good()* is true), then if the stream is tied to another output stream, `is.tie()->flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file ostream.tcc.

References std::ios_base::failbit, std::basic_ios< _CharT, _Traits >::good(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::tie().

4.625.2.2 template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::sentry::∼sentry ( ) [inline]

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 428 of file ostream.

References std::ios_base::badbit, std::uncaught_exception(), and std::ios_base::unitbuf.

4.625.3 Member Function Documentation

4.625.3.1 template<typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits >::sentry::operator bool ( ) const [inline], [explicit]

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

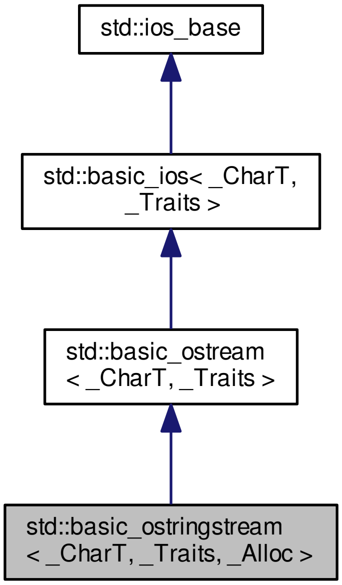
Definition at line 449 of file ostream.

The documentation for this class was generated from the following files:

- ostream
- ostream.tcc

4.626 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference

Inheritance diagram for std::basic_ostringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream < char_type, traits_type > **__ostream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf < _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ostringstream (ios_base::openmode __mode=ios_base::out)
- basic_ostringstream (const __string_type &__str, ios_base::openmode __mode=ios_base::out)
- ∼basic_ostringstream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)

- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (const void *__p)
- __ostream_type & operator<< (__streambuf_type *__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (pos_type)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type str () const
- void str (const __string_type &__s)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

  - __ostream_type & put (char_type __c)
  - void _M_write (const char_type ∗__s, streamsize __n)
  - __ostream_type & write (const char_type ∗__s, streamsize __n)

- operator void ∗ () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific

- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
  __ostream_type & **_M_insert** (_ValueT __v)
- void init (basic_streambuf< _CharT, _Traits > ∗__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.626.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>class std::basic_ostringstream< _CharT, _Traits, _Alloc >

Controlling output for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports writing to objects of type std::basic_string, using the inherited functions from std::basic_ostream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

Definition at line 387 of file sstream.

4.626.2 Member Typedef Documentation

4.626.2.1 template<typename _CharT, typename _Traits> typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits >::__num_get_type `[inherited]`

These are non-standard types.

Definition at line 90 of file basic_ios.h.

4.626.2.2 typedef void(* std::ios_base::event_callback)(event __e, ios_base &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.626.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal

- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.626.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.626.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.626.2.6 typedef _Ios_Seekdir std::ios_base::seekdir [inherited]

This is an enumerated type.

`_Ios_Seekdir is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.626.3 Member Enumeration Documentation

4.626.3.1 enum **std::ios_base::event** [inherited]

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.626.4 Constructor & Destructor Documentation

4.626.4.1 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_ostringstream**< _CharT, _Traits, _Alloc >**::basic_ostringstream (ios_base::openmode** *__mode* = **ios_base::out)** [inline], [explicit]

Default constructor starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 423 of file sstream.

References std::basic_ios< _CharT, _Traits >::init().

4.626.4.2 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_ostringstream**< _CharT, _Traits, _Alloc >**::basic_ostringstream (** const **__string_type &** *__str,* **ios_base::openmode** *__mode* = **ios_base::out)** [inline], [explicit]

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using *str* and `mode|out`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 441 of file sstream.

References std::basic_ios< _CharT, _Traits >::init().

4.626.4.3 template<typename _CharT, typename _Traits, typename _Alloc> std::basic_ostringstream< _CharT, _Traits, _Alloc >::∼basic_ostringstream ( ) `[inline]`

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 452 of file sstream.

4.626.5 Member Function Documentation

4.626.5.1 const locale& std::ios_base::_M_getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.626.5.2 template<typename _CharT, typename _Traits> void std::basic_ostream< _CharT, _Traits >::_M_write (const char_type ∗ *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Core write functionality, without sentry.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Definition at line 311 of file ostream.

Referenced by std::basic_ostream< _CharT, _Traits >::write().

4.626.5.3 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::bad ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.626.5.4 template<typename _CharT , typename _Traits > void basic_ios< _CharT, _Traits >::clear (iostate *__state* = goodbit) [inherited]

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.626.5.5 template<typename _CharT , typename _Traits > basic_ios< _CharT, _Traits > & basic_ios< _CharT, _Traits >::copyfmt (const basic_ios< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.626.5.6 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.626.5.7 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.626.5.8 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate __except) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.626.5.9 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::fail ( ) const [inline], [inherited]

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.626.5.10 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill ( ) const [inline], [inherited]

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.626.5.11 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::fill (char_type __ch) [inline], [inherited]

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.626.5.12 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.626.5.13 fmtflags std::ios_base::flags (fmtflags __fmtfl) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.626.5.14 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::flush ( ) `[inherited]`

Synchronizing the stream buffer.

Returns

∗this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::flush().

4.626.5.15 locale std::ios_base::getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.626.5.16 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::good ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.626.5.17 template<typename _CharT , typename _Traits > locale basic_ios< _CharT, _Traits >::imbue (const locale & __loc) [inherited]

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.626.5.18 template<typename _CharT, typename _Traits> void basic_ios< _CharT, _Traits >::init (basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.626.5.19 long& std::ios_base::iword (int __ix) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.626.5.20 template<typename _CharT, typename _Traits> char **std::basic_ios**< _CharT, _Traits >::narrow (**char_type** *__c*, char *__dfault*) const `[inline]`, `[inherited]`

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Definition at line 424 of file basic_ios.h.

4.626.5.21 template<typename _CharT, typename _Traits> **std::basic_ios**< _CharT, _Traits >::operator void ∗ ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.626.5.22 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.626.5.23 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ostream_type** &(∗)(**__ostream_type** &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 108 of file ostream.

4.626.5.24 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ios_type** &(∗)(**__ios_type** &) *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 117 of file ostream.

4.626.5.25 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (ios_base &(*)(ios_base &) __pf) [inline], [inherited]

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 127 of file ostream.

4.626.5.26 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

4.626.5.27 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.626.5.28 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.626.5.29 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (short __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.626.5.30 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.626.5.31 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.626.5.32 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

4.626.5.33 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

4.626.5.34 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

4.626.5.35 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

4.626.5.36 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float *__f*) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

4.626.5.37 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long double *__f*) [inline], [inherited]

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.626.5.38 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * *__p*) [inline], [inherited]

Pointer arithmetic inserters.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.626.5.39 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * __sb) `[inherited]`

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.626.5.40 streamsize std::ios_base::precision ( ) const `[inline]`, `[inherited]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.626.5.41 streamsize std::ios_base::precision (streamsize __prec) `[inline]`, `[inherited]`

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.626.5.42 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::put (char_type __c) [inherited]

Simple insertion.

Parameters

__c	The character to insert.

Returns

∗this

Tries to insert `__c`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.626.5.43 void∗& std::ios_base::pword (int __ix) [inline], [inherited]

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.626.5.44 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > ∗ basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > ∗ __sb) [inherited]

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.626.5.45 template<typename _CharT, typename _Traits, typename _Alloc> __stringbuf_type* **std::basic_ostringstream**< _CharT, _Traits, _Alloc >::rdbuf ( ) const `[inline]`

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

Definition at line 463 of file sstream.

4.626.5.46 template<typename _CharT, typename _Traits> **iostate std::basic_ios**< _CharT, _Traits >::rdstate ( ) const `[inline]`, `[inherited]`

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.626.5.47 void std::ios_base::register_callback (**event_callback** *__fn*, int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.626.5.48 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::seekp (pos_type __pos) [inherited]

Changing the current write position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.626.5.49 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::seekp (off_type __off, ios_base::seekdir __dir) [inherited]

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.626.5.50 fmtflags std::ios_base::setf (fmtflags __fmtfl) [inline], [inherited]

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.626.5.51 fmtflags std::ios_base::setf (fmtflags *__fmtfl,* fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.626.5.52 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.626.5.53 template<typename _CharT, typename _Traits, typename _Alloc> __string_type std::basic_ostringstream< _CharT, _Traits, _Alloc >::str ( ) const `[inline]`

Copying out the string buffer.

Returns

`rdbuf()->str()`

Definition at line 471 of file sstream.

Referenced by std::operator<<().

4.626.5.54 template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (const __string_type & *__s*) [inline]

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Calls `rdbuf()->str(s).`

Definition at line 481 of file sstream.

4.626.5.55 static bool std::ios_base::sync_with_stdio (bool *__sync* = true) [static], [inherited]

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.626.5.56 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp ( ) [inherited]

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out).`

Definition at line 237 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.626.5.57 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie ( ) const [inline], [inherited]

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.626.5.58 template<typename _CharT, typename _Traits> basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (basic_ostream< _CharT, _Traits > * *__tiestr*) `[inline]`, `[inherited]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.626.5.59 void std::ios_base::unsetf (fmtflags *__mask*) `[inline]`, `[inherited]`

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.626.5.60 template<typename _CharT, typename _Traits> char_type std::basic_ios< _CharT, _Traits >::widen (char *__c*) const `[inline]`, `[inherited]`

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.626.5.61 streamsize std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.626.5.62 streamsize std::ios_base::width (streamsize __wide) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.626.5.63 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::write (const char_type ∗ __s, streamsize __n) [inherited]

Character string insertion.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Returns

∗this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted

- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic_ostream< _CharT, _Traits >::_M_write(), and std::ios_base::badbit.

4.626.5.64 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.626.6 Member Data Documentation

4.626.6.1 const fmtflags std::ios_base::adjustfield [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.626.6.2 const openmode std::ios_base::app [static], [inherited]

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.626.6.3 const openmode std::ios_base::ate [static], [inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.626.6.4 const iostate std::ios_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.626.6.5 const **fmtflags** std::ios_base::basefield `[static]`, `[inherited]`

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.626.6.6 const **seekdir** std::ios_base::beg `[static]`, `[inherited]`

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.626.6.7 const **openmode** std::ios_base::binary `[static]`, `[inherited]`

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.626.6.8 const **fmtflags** std::ios_base::boolalpha `[static]`, `[inherited]`

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.626.6.9 const **seekdir** std::ios_base::cur `[static]`, `[inherited]`

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.626.6.10 const **fmtflags** std::ios_base::dec `[static]`, `[inherited]`

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.626.6.11 const seekdir std::ios_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.626.6.12 const iostate std::ios_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.626.6.13 const iostate std::ios_base::failbit [static], [inherited]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.626.6.14 const fmtflags std::ios_base::fixed [static], [inherited]

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.626.6.15 const fmtflags std::ios_base::floatfield [static], [inherited]

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.626.6.16 const iostate std::ios_base::goodbit [static], [inherited]

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.626.6.17 const **fmtflags** std::ios_base::hex `[static]`, `[inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.626.6.18 const **openmode** std::ios_base::in `[static]`, `[inherited]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.626.6.19 const **fmtflags** std::ios_base::internal `[static]`, `[inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.626.6.20 const **fmtflags** std::ios_base::left `[static]`, `[inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.626.6.21 const **fmtflags** std::ios_base::oct `[static]`, `[inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.626.6.22 const openmode std::ios_base::out `[static]`, `[inherited]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.626.6.23 const fmtflags std::ios_base::right `[static]`, `[inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.626.6.24 const fmtflags std::ios_base::scientific `[static]`, `[inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.626.6.25 const fmtflags std::ios_base::showbase `[static]`, `[inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.626.6.26 const fmtflags std::ios_base::showpoint `[static]`, `[inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.626.6.27 const fmtflags std::ios_base::showpos `[static]`, `[inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.626.6.28 const fmtflags std::ios_base::skipws `[static]`, `[inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.626.6.29 const openmode std::ios_base::trunc `[static]`, `[inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.626.6.30 const fmtflags std::ios_base::unitbuf [static], [inherited]

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.626.6.31 const fmtflags std::ios_base::uppercase [static], [inherited]

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- sstream

4.627 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference

Public Types

- typedef
  regex_constants::syntax_option_type **flag_type**
- typedef traits_type::locale_type **locale_type**
- typedef traits_type::string_type **string_type**
- typedef _Rx_traits **traits_type**
- typedef _Ch_type **value_type**

Public Member Functions

- basic_regex ()
- basic_regex (const _Ch_type ∗__p, flag_type __f=ECMAScript)
- basic_regex (const _Ch_type ∗__p, std::size_t __len, flag_type __f)
- basic_regex (const basic_regex &__rhs)
- basic_regex (const basic_regex &&__rhs) noexcept
- template<typename _Ch_traits , typename _Ch_alloc >
  basic_regex (const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, flag_type __f=ECMAScript)
- template<typename _InputIterator >
  basic_regex (_InputIterator __first, _InputIterator __last, flag_type __f=ECMAScript)
- basic_regex (initializer_list< _Ch_type > __l, flag_type __f=ECMAScript)
- ∼basic_regex ()
- const __detail::_AutomatonPtr & **_M_get_automaton** () const
- basic_regex & assign (const basic_regex &__rhs)
- basic_regex & assign (basic_regex &&__rhs) noexcept
- basic_regex & assign (const _Ch_type ∗__p, flag_type __flags=ECMAScript)
- basic_regex & assign (const _Ch_type ∗__p, std::size_t __len, flag_type __flags)
- template<typename _Ch_typeraits , typename _Alloc >
  basic_regex & assign (const basic_string< _Ch_type, _Ch_typeraits, _Alloc > &__s, flag_type __flags=ECMAScript)

- template<typename _InputIterator >
  basic_regex & assign (_InputIterator __first, _InputIterator __last, flag_type __flags=ECMAScript)
- basic_regex & assign (initializer_list< _Ch_type > __l, flag_type __flags=ECMAScript)
- flag_type flags () const
- locale_type getloc () const
- locale_type imbue (locale_type __loc)
- unsigned int mark_count () const
- basic_regex & operator= (const basic_regex &__rhs)
- basic_regex & operator= (basic_regex &&__rhs) noexcept
- basic_regex & operator= (const _Ch_type *__p)
- template<typename _Ch_typeraits , typename _Alloc >
  basic_regex & operator= (const basic_string< _Ch_type, _Ch_typeraits, _Alloc > &__s)
- void swap (basic_regex &__rhs)

Static Public Attributes

Constants

std [28.8.1](1)

- static constexpr flag_type **icase**
- static constexpr flag_type **nosubs**
- static constexpr flag_type **optimize**
- static constexpr flag_type **collate**
- static constexpr flag_type **ECMAScript**
- static constexpr flag_type **basic**
- static constexpr flag_type **extended**
- static constexpr flag_type **awk**
- static constexpr flag_type **grep**
- static constexpr flag_type **egrep**

Protected Attributes

- __detail::_AutomatonPtr **_M_automaton**
- flag_type **_M_flags**
- _Rx_traits **_M_traits**

4.627.1 Detailed Description

template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>class std::basic_regex< _Ch_type, _Rx_traits >

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 335 of file regex.h.

4.627.2 Constructor & Destructor Documentation

4.627.2.1 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits >::basic_regex ( ) `[inline]`

Constructs a basic regular expression that does not match any character sequence.

Definition at line 367 of file regex.h.

4.627.2.2 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::basic_regex**< _Ch_type, _Rx_traits >::**basic_regex** (const _Ch_type * *__p,* flag_type *__f* = `ECMAScript`) `[inline]`, `[explicit]`

Constructs a basic regular expression from the sequence [__p, __p + char_traits<_Ch_type>::length(__p)) interpreted according to the flags in __`f`.

Parameters

__p	A pointer to the start of a C-style null-terminated string containing a regular expression.
__f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __`p` is not a valid regular expression.

Definition at line 385 of file regex.h.

4.627.2.3 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::basic_regex**< _Ch_type, _Rx_traits >::**basic_regex** (const _Ch_type * *__p,* std::size_t *__len,* flag_type *__f*) `[inline]`

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in `f`.

Parameters

__p	A pointer to the start of a string containing a regular expression.
__len	The length of the string containing the regular expression.
__f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __`p` is not a valid regular expression.

Definition at line 403 of file regex.h.

4.627.2.4 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::basic_regex**< _Ch_type, _Rx_traits >::**basic_regex** (const **basic_regex**< _Ch_type, _Rx_traits > & *__rhs*) `[inline]`

Copy-constructs a basic regular expression.

Parameters

__rhs	A `regex` object.

Definition at line 413 of file regex.h.

4.627.2.5 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::basic_regex**< _Ch_type, _Rx_traits >::**basic_regex** (const **basic_regex**< _Ch_type, _Rx_traits > && *__rhs*) `[inline]`, `[noexcept]`

Move-constructs a basic regular expression.

Parameters

__rhs	A `regex` object.

Definition at line 423 of file regex.h.

4.627.2.6 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_traits , typename _Ch_alloc > std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s, flag_type __f = `ECMAScript`) `[inline]`, `[explicit]`

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

Parameters

__s	A string containing a regular expression.
__f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __`s` is not a valid regular expression.

Definition at line 439 of file regex.h.

4.627.2.7 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _InputIterator > std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (_InputIterator __first, _InputIterator __last, flag_type __f = `ECMAScript`) `[inline]`

Constructs a basic regular expression from the range [first, last) interpreted according to the flags in `f`.

Parameters

__first	The start of a range containing a valid regular expression.
__last	The end of a range containing a valid regular expression.
__f	The format flags of the regular expression.

Exceptions

regex_error	if [__first, __last) is not a valid regular expression.

Definition at line 461 of file regex.h.

4.627.2.8 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (initializer_list< _Ch_type > __l, flag_type __f = `ECMAScript`) `[inline]`

Constructs a basic regular expression from an initializer list.

Parameters

__l	The initializer list.
__f	The format flags of the regular expression.

Exceptions

regex_error	if __`l` is not a valid regular expression.

Definition at line 475 of file regex.h.

4.627.2.9 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> std::basic_regex< _Ch_type, _Rx_traits >::∼basic_regex ( ) [inline]

Destroys a basic regular expression.

Definition at line 485 of file regex.h.

4.627.3 Member Function Documentation

4.627.3.1 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (const basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline]

the real assignment operator.

Parameters

__rhs	Another regular expression object.

Definition at line 531 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::swap().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::operator=().

4.627.3.2 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (basic_regex< _Ch_type, _Rx_traits > && __rhs) [inline], [noexcept]

The move-assignment operator.

Parameters

__rhs	Another regular expression object.

Definition at line 544 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::swap().

4.627.3.3 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (const _Ch_type ∗ __p, flag_type __flags = ECMAScript) [inline]

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

Parameters

__p	A pointer to a C-style null-terminated string containing a regular expression pattern.
__flags	Syntax option flags.

Exceptions

regex_error	if __p does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, ∗this remains unchanged.

Definition at line 565 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

4.627.3.4 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (const _Ch_type * __p, std::size_t __len, flag_type __flags) [inline]

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

Parameters

__p	A pointer to a C-style string containing a regular expression pattern.
__len	The length of the regular expression pattern string.
__flags	Syntax option flags.

Exceptions

regex_error	if p does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, *this remains unchanged.

Definition at line 582 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

4.627.3.5 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_typeraits , typename _Alloc > basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (const basic_string< _Ch_type, _Ch_typeraits, _Alloc > & __s, flag_type __flags = `ECMAScript`) [inline]

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

Parameters

__s	A string containing a regular expression pattern.
__flags	Syntax option flags.

Exceptions

regex_error	if __s does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, *this remains unchanged.

Definition at line 598 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::swap().

4.627.3.6 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _InputIterator > basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (_InputIterator __first, _InputIterator __last, flag_type __flags = `ECMAScript`) [inline]

Assigns a new regular expression to a regex object.

Parameters

__first	The start of a range containing a valid regular expression.
__last	The end of a range containing a valid regular expression.
__flags	Syntax option flags.

Exceptions

regex_error	if p does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, the object remains unchanged.

Definition at line 621 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

4.627.3.7 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (initializer_list< _Ch_type > *__l*, flag_type *__flags* = `ECMAScript`) `[inline]`

Assigns a new regular expression to a regex object.

Parameters

__l	An initializer list representing a regular expression.
__flags	Syntax option flags.

Exceptions

regex_error	if `__l` does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, the object remains unchanged.

Definition at line 637 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign().

4.627.3.8 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> flag_type std::basic_regex< _Ch_type, _Rx_traits >::flags ( ) const `[inline]`

Gets the flags used to construct the regular expression or in the last call to assign().

Definition at line 654 of file regex.h.

Referenced by std::basic_regex< _Ch_type, _Rx_traits >::operator=().

4.627.3.9 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type, _Rx_traits >::getloc ( ) const `[inline]`

Gets the locale currently imbued in the regular expression object.

Definition at line 672 of file regex.h.

4.627.3.10 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> locale_type std::basic_regex< _Ch_type, _Rx_traits >::imbue (locale_type *__loc*) `[inline]`

Imbues the regular expression object with the given locale.

Parameters

__loc	A locale.

Definition at line 664 of file regex.h.

4.627.3.11 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> unsigned int **std::basic_regex**< _Ch_type, _Rx_traits >::mark_count ( ) const [inline]

Gets the number of marked subexpressions within the regular expression.

Definition at line 646 of file regex.h.

4.627.3.12 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **basic_regex**& **std::basic_regex**< _Ch_type, _Rx_traits >::operator= (const **basic_regex**< _Ch_type, _Rx_traits > & *__rhs*) [inline]

Assigns one regular expression to another.

Definition at line 492 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

4.627.3.13 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **basic_regex**& **std::basic_regex**< _Ch_type, _Rx_traits >::operator= (**basic_regex**< _Ch_type, _Rx_traits > && *__rhs*) [inline], [noexcept]

Move-assigns one regular expression to another.

Definition at line 499 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign().

4.627.3.14 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> **basic_regex**& **std::basic_regex**< _Ch_type, _Rx_traits >::operator= (const _Ch_type * *__p*) [inline]

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

__p	A pointer to the start of a null-terminated C-style string containing a regular expression.

Definition at line 510 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign(), and std::basic_regex< _Ch_type, _Rx_traits >::flags().

4.627.3.15 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> template<typename _Ch_typeraits , typename _Alloc > **basic_regex**& **std::basic_regex**< _Ch_type, _Rx_traits >::operator= (const **basic_string**< _Ch_type, _Ch_typeraits, _Alloc > & *__s*) [inline]

Replaces a regular expression with a new one constructed from a string.

Parameters

__s	A pointer to a string containing a regular expression.

Definition at line 521 of file regex.h.

References std::basic_regex< _Ch_type, _Rx_traits >::assign(), and std::basic_regex< _Ch_type, _Rx_traits >::flags().

4.627.3.16 template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>> void **std::basic_regex**< _Ch_type, _Rx_traits >::swap (**basic_regex**< _Ch_type, _Rx_traits > & *__rhs*) [inline]

Swaps the contents of two regular expression objects.

Parameters

__rhs	Another regular expression object.

Definition at line 682 of file regex.h.

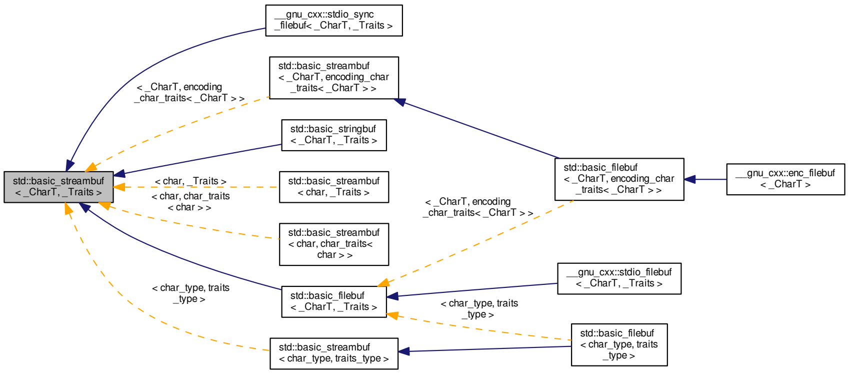
Referenced by std::basic_regex< _Ch_type, _Rx_traits >::assign(), and std::swap().

The documentation for this class was generated from the following file:

- regex.h

4.628 std::basic_streambuf< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::basic_streambuf< _CharT, _Traits >:



Public Types

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef traits_type::int_type int_type
- typedef traits_type::pos_type pos_type
- typedef traits_type::off_type off_type

- typedef basic_streambuf < char_type, traits_type > __streambuf_type

Public Member Functions

- virtual ∼basic_streambuf ()
- locale getloc () const
- streamsize in_avail ()
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type ∗__s, streamsize __n)

- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type ∗__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf ∗ pubsetbuf (char_type ∗__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- basic_streambuf ()
- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf
< char_type, _Traits > ∗ setbuf (char_type ∗, streamsize)
- void setg (char_type ∗__gbeg, char_type ∗__gnext, char_type ∗__gend)
- void setp (char_type ∗__pbeg, char_type ∗__pend)
- virtual streamsize showmanyc ()
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type ∗__s, streamsize __n)
- virtual streamsize xsputn (const char_type ∗__s, streamsize __n)

- char_type ∗ eback () const
- char_type ∗ gptr () const
- char_type ∗ egptr () const

- char_type ∗ pbase () const
- char_type ∗ pptr () const
- char_type ∗ epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type ∗ _M_in_beg
- char_type ∗ _M_in_cur
- char_type ∗ _M_in_end
- char_type ∗ _M_out_beg
- char_type ∗ _M_out_cur
- char_type ∗ _M_out_end

Friends

- template<bool _IsMove, typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, _CharT2 ∗ >::__type **__copy_move_a2** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 ∗)
- streamsize **__copy_streambufs_eof** (basic_streambuf ∗, basic_streambuf ∗, bool &)
- class **basic_ios< char_type, traits_type >**
- class **basic_istream< char_type, traits_type >**
- class **basic_ostream< char_type, traits_type >**
- template<typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, istreambuf_iterator
  < _CharT2 > >::__type **find** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)
- template<typename _CharT2 , typename _Traits2 , typename _Alloc >
  basic_istream< _CharT2,
  _Traits2 > & **getline** (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)
- class **istreambuf_iterator< char_type, traits_type >**
- template<typename _CharT2 , typename _Traits2 >
  basic_istream< _CharT2,
  _Traits2 > & **operator>>** (basic_istream< _CharT2, _Traits2 > &, _CharT2 ∗)
- template<typename _CharT2 , typename _Traits2 , typename _Alloc >
  basic_istream< _CharT2,
  _Traits2 > & **operator>>** (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)
- class **ostreambuf_iterator< char_type, traits_type >**

4.628.1 Detailed Description

template<typename _CharT, typename _Traits>class std::basic_streambuf< _CharT, _Traits >

The actual work of input and output (interface).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:
   - The controlled input sequence can be not readable.
   - The controlled output sequence can be not writable.
   - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.

   - The controlled sequences can support operations *directly* to or from associated sequences.
   - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.

2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:

   - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
   - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
   - the *end pointer*, or first element address beyond the end of the array (called *xend* here).

3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:

   - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
   - If *xnext* is not a null pointer and *xnext* $<$ *xend* for an output sequence, then a *write position* is available. In this case, $*$*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
   - If *xnext* is not a null pointer and *xbeg* $<$ *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext*[-1] shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
   - If *xnext* is not a null pointer and *xnext*$<$ *xend* for an input sequence, then a *read position* is available. In this case, $*$*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 120 of file streambuf.

4.628.2 Member Typedef Documentation

4.628.2.1 template<typename _CharT, typename _Traits> typedef **basic_streambuf<char_type, traits_type>** **std::basic_streambuf< _CharT, _Traits >::__streambuf_type**

This is a non-standard type.

Definition at line 138 of file streambuf.

4.628.2.2 template<typename _CharT, typename _Traits> typedef _CharT **std::basic_streambuf< _CharT, _Traits >::char_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 129 of file streambuf.

4.628.2.3 template<typename _CharT, typename _Traits> typedef traits_type::int_type **std::basic_streambuf< _CharT, _Traits >::int_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 131 of file streambuf.

4.628.2.4 template<typename _CharT, typename _Traits> typedef traits_type::off_type **std::basic_streambuf**< _CharT, _Traits >**::off_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 133 of file streambuf.

4.628.2.5 template<typename _CharT, typename _Traits> typedef traits_type::pos_type **std::basic_streambuf**< _CharT, _Traits >**::pos_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 132 of file streambuf.

4.628.2.6 template<typename _CharT, typename _Traits> typedef _Traits **std::basic_streambuf**< _CharT, _Traits >**::traits_type**

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 130 of file streambuf.

4.628.3 Constructor & Destructor Documentation

4.628.3.1 template<typename _CharT, typename _Traits> virtual **std::basic_streambuf**< _CharT, _Traits >::∼**basic_streambuf ( )** `[inline]`, `[virtual]`

Destructor deallocates no buffer space.

Definition at line 197 of file streambuf.

4.628.3.2 template<typename _CharT, typename _Traits> **std::basic_streambuf**< _CharT, _Traits >::**basic_streambuf ( )** `[inline]`, `[protected]`

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the basic_streambuf class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 463 of file streambuf.

4.628.4 Member Function Documentation

4.628.4.1 template<typename _CharT, typename _Traits> **char_type**∗ **std::basic_streambuf**< _CharT, _Traits >::eback ( ) const `[inline]`, `[protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence

- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sungetc().

4.628.4.2 template<typename _CharT, typename _Traits> **char_type∗ std::basic_streambuf**< _CharT, _Traits >::egptr ( ) const [inline], [protected]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.628.4.3 template<typename _CharT, typename _Traits> **char_type∗ std::basic_streambuf**< _CharT, _Traits >::epptr ( ) const [inline], [protected]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.628.4.4 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::gbump (int *__n*) [inline], [protected]

Moving the read position.

Parameters

__n	The delta by which to move.

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.628.4.5 template<typename _CharT, typename _Traits> **locale std::basic_streambuf**< _CharT, _Traits >::getloc ( ) const `[inline]`

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.628.4.6 template<typename _CharT, typename _Traits> **char_type* std::basic_streambuf**< _CharT, _Traits >::gptr ( ) const `[inline]`, `[protected]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.628.4.7 template<typename _CharT, typename _Traits> virtual void **std::basic_streambuf**< _CharT, _Traits >::imbue (const **locale** & *__loc*) `[inline]`, `[protected]`, `[virtual]`

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 576 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.628.4.8 template<typename _CharT, typename _Traits> **streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ( )** `[inline]`

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.628.4.9 template<typename _CharT, typename _Traits> virtual **int_type std::basic_streambuf< _CharT, _Traits >::overflow (int_type *__c* =** `traits_type::eof()` **)** `[inline]`, `[protected]`, `[virtual]`

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, and __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 768 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.628.4.10 template<typename _CharT, typename _Traits> virtual **int_type std::basic_streambuf**< _CharT, _Traits >::pbackfail (**int_type** *__c* = `traits_type::eof()`) `[inline]`, `[protected]`, `[virtual]`

Tries to back up the input sequence.

Parameters

__c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, and __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 724 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sungetc().

4.628.4.11 template<typename _CharT, typename _Traits> **char_type**∗ **std::basic_streambuf**< _CharT, _Traits >::pbase ( ) const `[inline]`, `[protected]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.628.4.12 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::pbump (int *__n*) `[inline]`, `[protected]`

Moving the write position.

Parameters

__n	The delta by which to move.

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.628.4.13 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::pptr ( ) const [inline], [protected]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.628.4.14 template<typename _CharT, typename _Traits> locale std::basic_streambuf< _CharT, _Traits >::pubimbue (const locale & *__loc*) [inline]

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

Definition at line 209 of file streambuf.

4.628.4.15 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (off_type *__off*, ios_base::seekdir *__way*, ios_base::openmode *__mode* = ios_base::in | ios_base::out) [inline]

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.628.4.16 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type *__sp*, ios_base::openmode *__mode* = ios_base::in | ios_base::out) [inline]

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

4.628.4.17 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (char_type * *__s*, streamsize *__n*) [inline]

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

4.628.4.18 template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline]

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::sync().

4.628.4.19 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline]

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::istreambuf_iterator< _CharT, _Traits >::operator++(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.628.4.20 template<typename _CharT, typename _Traits> virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (off_type , ios_base::seekdir , ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 602 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubseekoff().

4.628.4.21 template<typename _CharT, typename _Traits> virtual **pos_type std::basic_streambuf**< _CharT, _Traits >::seekpos (**pos_type** , **ios_base::openmode** = ios_base::in | ios_base::out) `[inline]`, `[protected]`, `[virtual]`

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 614 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubseekpos().

4.628.4.22 template<typename _CharT, typename _Traits> virtual **basic_streambuf<char_type,_Traits>*** **std::basic_streambuf**< _CharT, _Traits >::setbuf (**char_type** * , **streamsize**) `[inline]`, `[protected]`, `[virtual]`

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 591 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubsetbuf().

4.628.4.23 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setg (**char_type** * *__gbeg*, **char_type** * *__gnext*, **char_type** * *__gend*) `[inline]`, `[protected]`

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Definition at line 509 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.628.4.24 template<typename _CharT, typename _Traits> void std::basic_streambuf< _CharT, _Traits >::setp (char_type ∗ *__pbeg,* char_type ∗ *__pend*) `[inline]`, `[protected]`

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Definition at line 555 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.628.4.25 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sgetc ( ) `[inline]`

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.628.4.26 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (char_type ∗ *__s,* streamsize *__n*) `[inline]`

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.628.4.27 template<typename _CharT, typename _Traits> virtual **streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ( )** `[inline]`, `[protected]`, `[virtual]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.
The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*
The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and std::basic_stringbuf< _CharT, _Traits, _Alloc >.

Definition at line 649 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::in_avail().

4.628.4.28 template<typename _CharT, typename _Traits> **int_type std::basic_streambuf< _CharT, _Traits >::snextc ( )** `[inline]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.628.4.29 template<typename _CharT, typename _Traits> **int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type *__c*)** `[inline]`

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.

Definition at line 372 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::putback().

4.628.4.30 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c) [inline]

Entry point for all single-character output functions.

Parameters

__c	A character to output.

Returns

__c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *__c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 424 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), and std::ostreambuf_iterator< _CharT, _Traits >::operator=().

4.628.4.31 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * __s, streamsize __n) [inline]

Entry point for all single-character output functions.

Parameters

__s	A buffer read area.
__n	A count.

One of two public output functions.

Returns xsputn(__s,__n). The effect is to write *__s*[0] through *__s*[__n-1] to the output sequence, if possible.

Definition at line 450 of file streambuf.

4.628.4.32 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sungetc ( ) [inline]

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

4.628.4.33 template<typename _CharT, typename _Traits> virtual int **std::basic_streambuf**< _CharT, _Traits >::sync (void) `[inline]`, `[protected]`, `[virtual]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 627 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubsync().

4.628.4.34 template<typename _CharT, typename _Traits> virtual **int_type std::basic_streambuf**< _CharT, _Traits >::uflow ( ) `[inline]`, `[protected]`, `[virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 700 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc().

4.628.4.35 template<typename _CharT, typename _Traits> virtual **int_type std::basic_streambuf**< _CharT, _Traits >::underflow ( ) `[inline]`, `[protected]`, `[virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html`

Note

Base class version does nothing, returns eof().

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, and __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 687 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sgetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.628.4.36 template<typename _CharT , typename _Traits > streamsize basic_streambuf< _CharT, _Traits >::xsgetn (char_type * __s, streamsize __n) `[protected]`, `[virtual]`

Multiple character extraction.

Parameters

__s	A buffer area.
__n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 46 of file streambuf.tcc.

References std::min().

Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

4.628.4.37 template<typename _CharT , typename _Traits > streamsize basic_streambuf< _CharT, _Traits >::xsputn (const char_type * __s, streamsize __n) `[protected]`, `[virtual]`

Multiple character insertion.

Parameters

__s	A buffer area.
__n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 80 of file streambuf.tcc.

References std::min().

Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

4.628.5 Member Data Documentation

4.628.5.1 template<typename _CharT, typename _Traits> **locale std::basic_streambuf**< _CharT, _Traits >::_M_buf_locale `[protected]`

Current locale setting.

Definition at line 192 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf(), std::basic_streambuf< char_type, traits_type >::getloc(), and std::basic_streambuf< char_type, traits_type >::pubimbue().

4.628.5.2 template<typename _CharT, typename _Traits> **char_type∗ std::basic_streambuf**< _CharT, _Traits >::_M_in_beg `[protected]`

Start of get area.

Definition at line 184 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::eback(), and std::basic_streambuf< char_type, traits_type >::setg().

4.628.5.3 template<typename _CharT, typename _Traits> **char_type∗ std::basic_streambuf**< _CharT, _Traits >::_M_in_cur `[protected]`

Current read area.

Definition at line 185 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::gbump(), std::basic_streambuf< char_type, traits_type >::gptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.628.5.4 template<typename _CharT, typename _Traits> **char_type∗ std::basic_streambuf**< _CharT, _Traits >::_M_in_end `[protected]`

End of get area.

Definition at line 186 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::egptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.628.5.5 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg `[protected]`

Start of put area.

Definition at line 187 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbase(), and std::basic_streambuf< char_type, traits_type >::setp().

4.628.5.6 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur `[protected]`

Current put area.

Definition at line 188 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbump(), std::basic_streambuf< char_type, traits_type >::pptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.628.5.7 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end `[protected]`

End of put area.

Definition at line 189 of file streambuf.

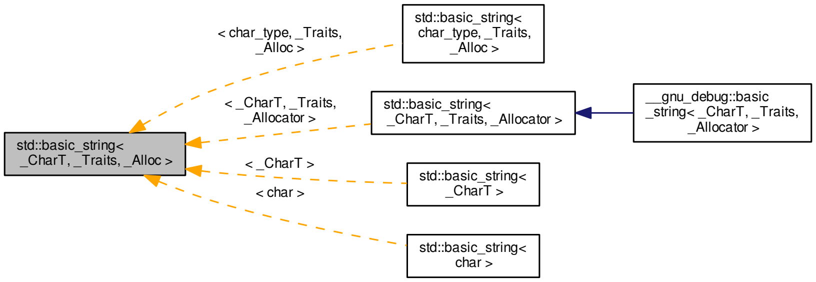
Referenced by std::basic_streambuf< char_type, traits_type >::epptr(), and std::basic_streambuf< char_type, traits_type >::setp().

The documentation for this class was generated from the following files:

- streambuf
- streambuf.tcc

4.629 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

Inheritance diagram for std::basic_string< _CharT, _Traits, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef
  __gnu_cxx::__normal_iterator
  < const_pointer, basic_string > **const_iterator**
- typedef
  _CharT_alloc_type::const_pointer **const_pointer**
- typedef
  _CharT_alloc_type::const_reference **const_reference**
- typedef std::reverse_iterator
  < const_iterator > **const_reverse_iterator**
- typedef
  _CharT_alloc_type::difference_type **difference_type**
- typedef
  __gnu_cxx::__normal_iterator
  < pointer, basic_string > **iterator**
- typedef _CharT_alloc_type::pointer **pointer**
- typedef
  _CharT_alloc_type::reference **reference**
- typedef std::reverse_iterator
  < iterator > **reverse_iterator**
- typedef
  _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- basic_string ()
- basic_string (const _Alloc &__a)
- basic_string (const basic_string &__str)
- basic_string (const basic_string &__str, size_type __pos, size_type __n=npos)
- basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- basic_string (const _CharT ∗__s, size_type __n, const _Alloc &__a=_Alloc())
- basic_string (const _CharT ∗__s, const _Alloc &__a=_Alloc())
- basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- basic_string (basic_string &&__str) noexcept
- basic_string (initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- template<class _InputIterator >
  basic_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- ∼basic_string () noexcept
- template<typename _InIterator >
  _CharT ∗ **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)
- basic_string & append (const basic_string &__str)
- basic_string & append (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & append (const _CharT ∗__s, size_type __n)
- basic_string & append (const _CharT ∗__s)
- basic_string & append (size_type __n, _CharT __c)
- basic_string & append (initializer_list< _CharT > __l)
- template<class _InputIterator >
  basic_string & append (_InputIterator __first, _InputIterator __last)

- basic_string & assign (const basic_string &__str)
- basic_string & assign (basic_string &&__str)
- basic_string & assign (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & assign (const _CharT ∗__s, size_type __n)
- basic_string & assign (const _CharT ∗__s)
- basic_string & assign (size_type __n, _CharT __c)
- template<class _InputIterator >
  basic_string & assign (_InputIterator __first, _InputIterator __last)
- basic_string & assign (initializer_list< _CharT > __l)
- const_reference at (size_type __n) const
- reference at (size_type __n)
- reference back ()
- const_reference back () const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const _CharT ∗ c_str () const noexcept
- size_type capacity () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- int compare (const basic_string &__str) const
- int compare (size_type __pos, size_type __n, const basic_string &__str) const
- int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2) const
- int compare (const _CharT ∗__s) const
- int compare (size_type __pos, size_type __n1, const _CharT ∗__s) const
- int compare (size_type __pos, size_type __n1, const _CharT ∗__s, size_type __n2) const
- size_type copy (_CharT ∗__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- const _CharT ∗ data () const noexcept
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- basic_string & erase (size_type __pos=0, size_type __n=npos)
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- size_type find (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find (const _CharT ∗__s, size_type __pos=0) const
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_first_not_of (const _CharT ∗__s, size_type __pos=0) const
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_first_of (const _CharT ∗__s, size_type __pos=0) const
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const _CharT ∗__s, size_type __pos, size_type __n) const

- size_type find_last_not_of (const _CharT ∗__s, size_type __pos=npos) const
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type find_last_of (const _CharT ∗__s, size_type __pos=npos) const
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- void insert (iterator __p, size_type __n, _CharT __c)
- template<class _InputIterator >
  void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, initializer_list< _CharT > __l)
- basic_string & insert (size_type __pos1, const basic_string &__str)
- basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n)
- basic_string & insert (size_type __pos, const _CharT ∗__s, size_type __n)
- basic_string & insert (size_type __pos, const _CharT ∗__s)
- basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- iterator insert (iterator __p, _CharT __c)
- size_type length () const noexcept
- size_type max_size () const noexcept
- basic_string & operator+= (const basic_string &__str)
- basic_string & operator+= (const _CharT ∗__s)
- basic_string & operator+= (_CharT __c)
- basic_string & operator+= (initializer_list< _CharT > __l)
- basic_string & operator= (const basic_string &__str)
- basic_string & operator= (const _CharT ∗__s)
- basic_string & operator= (_CharT __c)
- basic_string & operator= (basic_string &&__str)
- basic_string & operator= (initializer_list< _CharT > __l)
- const_reference operator[] (size_type __pos) const
- reference operator[] (size_type __pos)
- void pop_back ()
- void push_back (_CharT __c)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2)
- basic_string & replace (size_type __pos, size_type __n1, const _CharT ∗__s, size_type __n2)
- basic_string & replace (size_type __pos, size_type __n1, const _CharT ∗__s)
- basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT ∗__s, size_type __n)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT ∗__s)
- basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator >
  basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, _CharT ∗__k1, _CharT ∗__k2)

- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT ∗__k1, const _CharT ∗__k2)
- basic_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- void reserve (size_type __res_arg=0)
- void resize (size_type __n, _CharT __c)
- void resize (size_type __n)
- size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const _CharT ∗__s, size_type __pos, size_type __n) const
- size_type rfind (const _CharT ∗__s, size_type __pos=npos) const
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- void shrink_to_fit ()
- size_type size () const noexcept
- basic_string substr (size_type __pos=0, size_type __n=npos) const
- void swap (basic_string &__s)

Static Public Attributes

- static const size_type npos

4.629.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>class std::basic_string< _CharT, _Traits, _Alloc >

Managing sequences of characters and character-like objects.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`. Of the `optional sequence requirements`, only `push_back`, `at`, and `array` access are supported.

Todo Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

Documentation? What's that? Nathan Myers `ncm@cantrip.org`.

A string looks like this:

```
                                     [_Rep]
                                     _M_length
[basic_string<char_type>]            _M_capacity
_M_dataplus                          _M_refcount
_M_p ---------------->               unnamed array of char_type
```

Where the _M_p points to the first character in the string, and you cast it to a pointer-to-_Rep and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: _Rep::_M_data(), and string::_M_rep(); and the allocation function which gets a block of raw bytes and with room enough and constructs a _Rep object at the front.

The reason you want _M_data pointing to the character array and not the _Rep is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the _Rep for the debugger to use, so users can check the actual string length.)

Note that the _Rep object is a POD so that you can have a static *empty string* _Rep object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string _Rep object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 112 of file basic_string.h.

4.629.2 Constructor & Destructor Documentation

4.629.2.1 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** ( ) `[inline]`

Default constructor creates an empty string.

Definition at line 437 of file basic_string.h.

Referenced by std::basic_string< char >::substr().

4.629.2.2 template<typename _CharT , typename _Traits , typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** (const _Alloc & *__a*) `[explicit]`

Construct an empty string using allocator *a*.

Definition at line 178 of file basic_string.tcc.

4.629.2.3 template<typename _CharT , typename _Traits , typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** (const **basic_string**< _CharT, _Traits, _Alloc > & *__str*)

Construct string with copy of value of *str*.

Parameters

__str	Source string.

Definition at line 170 of file basic_string.tcc.

4.629.2.4 template<typename _CharT , typename _Traits , typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** (const **basic_string**< _CharT, _Traits, _Alloc > & *__str*, size_type *__pos*, size_type *__n* = **npos**)

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy (default remainder).

Definition at line 184 of file basic_string.tcc.

4.629.2.5 template<typename _CharT , typename _Traits , typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const basic_string< _CharT, _Traits, _Alloc > & *__str,* size_type *__pos,* size_type *__n,* const _Alloc & *__a*)

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

Definition at line 194 of file basic_string.tcc.

4.629.2.6 template<typename _CharT, typename _Traits , typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const _CharT ∗ *__s,* size_type *__n,* const _Alloc & *__a* = `_Alloc()`)

Construct string initialized by a character array.

Parameters

__s	Source character array.
__n	Number of characters to copy.
__a	Allocator to use (default is default allocator).

NB: *__s* must have at least *__n* characters, '\0' has no special meaning.

Definition at line 206 of file basic_string.tcc.

4.629.2.7 template<typename _CharT, typename _Traits , typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (const _CharT ∗ *__s,* const _Alloc & *__a* = `_Alloc()`)

Construct string as copy of a C string.

Parameters

__s	Source C string.
__a	Allocator to use (default is default allocator).

Definition at line 213 of file basic_string.tcc.

4.629.2.8 template<typename _CharT, typename _Traits , typename _Alloc> std::basic_string< _CharT, _Traits, _Alloc >::basic_string (size_type *__n,* _CharT *__c,* const _Alloc & *__a* = `_Alloc()`)

Construct string as multiple characters.

Parameters

__n	Number of characters.
__c	Character to use.
__a	Allocator to use (default is default allocator).

Definition at line 220 of file basic_string.tcc.

4.629.2.9 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** (**basic_string**< _CharT, _Traits, _Alloc > && *__str*) [inline], [noexcept]

Move construct string.

Parameters

__str	Source string.

The newly-created string contains the exact contents of *__str*. *__str* is a valid, but unspecified string.

Definition at line 507 of file basic_string.h.

4.629.2.10 template<typename _CharT, typename _Traits , typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** (**initializer_list**< _CharT > *__l*, const _Alloc & *__a* = _Alloc())

Construct string from an initializer list.

Parameters

__l	std::initializer_list of characters.
__a	Allocator to use (default is default allocator).

Definition at line 235 of file basic_string.tcc.

4.629.2.11 template<typename _CharT , typename _Traits , typename _Alloc> template<typename _InputIterator > **std::basic_string**< _CharT, _Traits, _Alloc >::**basic_string** (_InputIterator *__beg*, _InputIterator *__end*, const _Alloc & *__a* = _Alloc())

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

Definition at line 228 of file basic_string.tcc.

4.629.2.12 template<typename _CharT, typename _Traits, typename _Alloc> **std::basic_string**< _CharT, _Traits, _Alloc >::∼**basic_string** ( ) [inline], [noexcept]

Destroy the string instance.

Definition at line 538 of file basic_string.h.

4.629.3 Member Function Documentation

4.629.3.1 template<typename _CharT , typename _Traits , typename _Alloc > **basic_string**< _CharT, _Traits, _Alloc > & **std::basic_string**< _CharT, _Traits, _Alloc >::append (const **basic_string**< _CharT, _Traits, _Alloc > & *__str*)

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Definition at line 325 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::size(), and std::size().

Referenced by std::basic_string< char >::append(), std::collate< _CharT >::do_transform(), std::operator+(), std::basic_string< char >::operator+=(), and std::operator>>().

4.629.3.2 template<typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (const basic_string< _CharT, _Traits, _Alloc > & *__str*, size_type *__pos*, size_type *__n*)

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

Definition at line 342 of file basic_string.tcc.

References std::size().

4.629.3.3 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (const _CharT ∗ *__s*, size_type *__n*)

Append a C substring.

Parameters

__s	The C string to append.
__n	The number of characters to append.

Returns

Reference to this string.

Definition at line 298 of file basic_string.tcc.

References std::size().

4.629.3.4 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (const _CharT * __s) [inline]

Append a C string.

Parameters

__s	The C string to append.

Returns

Reference to this string.

Definition at line 1006 of file basic_string.h.

4.629.3.5 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (size_type __n, _CharT __c)

Append multiple characters.

Parameters

__n	The number of characters to append.
__c	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

Definition at line 281 of file basic_string.tcc.

References std::size().

4.629.3.6 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (initializer_list< _CharT > __l) [inline]

Append an initializer_list of characters.

Parameters

__l	The initializer_list of characters to append.

Returns

Reference to this string.

Definition at line 1030 of file basic_string.h.

Referenced by std::basic_string< char >::append().

4.629.3.7 template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > **basic_string**& **std::basic_string**< _CharT, _Traits, _Alloc >::append (_InputIterator *__first,* _InputIterator *__last*) `[inline]`

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [__first,__last) to this string.

Definition at line 1044 of file basic_string.h.

4.629.3.8 template<typename _CharT , typename _Traits , typename _Alloc > **basic_string**< _CharT, _Traits, _Alloc > & **std::basic_string**< _CharT, _Traits, _Alloc >::assign (const **basic_string**< _CharT, _Traits, _Alloc > & *__str*)

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Definition at line 243 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::get_allocator().

Referenced by std::basic_string< char >::assign(), std::basic_string< char >::operator=(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.629.3.9 template<typename _CharT, typename _Traits, typename _Alloc> **basic_string**& **std::basic_string**< _CharT, _Traits, _Alloc >::assign (**basic_string**< _CharT, _Traits, _Alloc > && *__str*) `[inline]`

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

Definition at line 1079 of file basic_string.h.

4.629.3.10 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos, size_type __n) [inline]

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

Definition at line 1100 of file basic_string.h.

Referenced by std::basic_string< char >::assign().

4.629.3.11 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (const _CharT * __s, size_type __n)

Set value to a C substring.

Parameters

__s	The C string to use.
__n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *__n* characters of *__s*. If *__n* is is larger than the number of available characters in *__s*, the remainder of *__s* is used.

Definition at line 259 of file basic_string.tcc.

References std::size().

4.629.3.12 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (const _CharT * __s) [inline]

Set value to contents of a C string.

Parameters

__s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *__s*. The data is copied, so there is no dependence on *__s* once the function returns.

Definition at line 1128 of file basic_string.h.

4.629.3.13 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (size_type *__n*, _CharT *__c*) [inline]

Set value to multiple characters.

Parameters

__n	Length of the resulting string.
__c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to *__n* copies of character *__c*.

Definition at line 1144 of file basic_string.h.

4.629.3.14 template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (_InputIterator *__first*, _InputIterator *__last*) [inline]

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [__first,__last).

Definition at line 1157 of file basic_string.h.

4.629.3.15 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (initializer_list< _CharT > *__l*) [inline]

Set value to an initializer_list of characters.

Parameters

__l	The initializer_list of characters to assign.

Returns

Reference to this string.

Definition at line 1167 of file basic_string.h.

Referenced by std::basic_string< char >::assign().

4.629.3.16 template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (size_type __n) const [inline]

Provides access to the data contained in the string.

Parameters

__n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

Definition at line 864 of file basic_string.h.

4.629.3.17 template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::at (size_type __n) [inline]

Provides access to the data contained in the string.

Parameters

__n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

Definition at line 883 of file basic_string.h.

4.629.3.18 template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::back ( ) [inline]

Returns a read/write reference to the data at the last element of the string.

Definition at line 913 of file basic_string.h.

4.629.3.19 template<typename _CharT, typename _Traits, typename _Alloc> const_reference **std::basic_string**< _CharT, _Traits, _Alloc >::back ( ) const [inline]

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 921 of file basic_string.h.

4.629.3.20 template<typename _CharT, typename _Traits, typename _Alloc> iterator **std::basic_string**< _CharT, _Traits, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 605 of file basic_string.h.

Referenced by std::basic_string< char >::crend(), std::regex_match(), std::regex_replace(), std::regex_search(), and std::basic_string< char >::rend().

4.629.3.21 template<typename _CharT, typename _Traits, typename _Alloc> const_iterator **std::basic_string**< _CharT, _Traits, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 616 of file basic_string.h.

4.629.3.22 template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* **std::basic_string**< _CharT, _Traits, _Alloc >::c_str ( ) const [inline], [noexcept]

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1800 of file basic_string.h.

Referenced by std::collate< _CharT >::do_compare(), std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::collate< _CharT >::do_transform(), std::basic_filebuf< char_type, traits_type >::open(), and std::operator==().

4.629.3.23 template<typename _CharT, typename _Traits, typename _Alloc> size_type **std::basic_string**< _CharT, _Traits, _Alloc >::capacity ( ) const [inline], [noexcept]

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 776 of file basic_string.h.

Referenced by std::basic_string< char >::push_back(), and std::basic_string< char >::shrink_to_fit().

4.629.3.24 template<typename _CharT, typename _Traits, typename _Alloc> const_iterator **std::basic_string**< _CharT, _Traits, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 680 of file basic_string.h.

4.629.3.25 template<typename _CharT, typename _Traits, typename _Alloc> const_iterator **std::basic_string**< _CharT, _Traits, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 688 of file basic_string.h.

4.629.3.26 template<typename _CharT, typename _Traits, typename _Alloc> void **std::basic_string**< _CharT, _Traits, _Alloc >::clear ( ) `[inline]`, `[noexcept]`

Erases the string, making it empty.

Definition at line 803 of file basic_string.h.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf().

4.629.3.27 template<typename _CharT, typename _Traits, typename _Alloc> int **std::basic_string**< _CharT, _Traits, _Alloc >::compare (const **basic_string**< _CharT, _Traits, _Alloc > & *__str*) const `[inline]`

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2225 of file basic_string.h.

Referenced by std::sub_match< _Bi_iter >::compare(), std::operator<(), std::operator<=(), std::operator==(), std::operator>(), and std::operator>=().

4.629.3.28 template<typename _CharT , typename _Traits , typename _Alloc > int **std::basic_string**< _CharT, _Traits, _Alloc >::compare (size_type *__pos*, size_type *__n*, const **basic_string**< _CharT, _Traits, _Alloc > & *__str*) const

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 916 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::data(), std::min(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

4.629.3.29 template<typename _CharT , typename _Traits , typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare (size_type *__pos1,* size_type *__n1,* const basic_string< _CharT, _Traits, _Alloc > & *__str,* size_type *__pos2,* size_type *__n2*) const

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 931 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::min().

4.629.3.30 template<typename _CharT, typename _Traits , typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare (const _CharT * *__s*) const

Compare to a C string.

Parameters

__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__s*, 0 if their values are equivalent, or > 0 if this string is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from *__s*. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 949 of file basic_string.tcc.

References std::min(), and std::size().

4.629.3.31 template<typename _CharT, typename _Traits , typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare (size_type *__pos,* size_type *__n1,* const _CharT * *__s*) const

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 964 of file basic_string.tcc.

References std::min().

4.629.3.32 template<typename _CharT, typename _Traits , typename _Alloc > int std::basic_string< _CharT, _Traits, _Alloc >::compare (size_type *__pos,* size_type *__n1,* const _CharT ∗ *__s,* size_type *__n2*) const

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: s must have at least n2 characters, '\0' has no special meaning.

Definition at line 980 of file basic_string.tcc.

References std::min().

4.629.3.33 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::copy (_CharT ∗ *__s,* size_type *__n,* size_type *__pos =* 0) const

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *__s*. If *__pos* is greater than size(), out_of_range is thrown.

Definition at line 723 of file basic_string.tcc.

4.629.3.34 template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 697 of file basic_string.h.

4.629.3.35 template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 706 of file basic_string.h.

4.629.3.36 template<typename _CharT, typename _Traits, typename _Alloc> const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::data ( ) const [inline], [noexcept]

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1810 of file basic_string.h.

Referenced by std::basic_string< char >::compare(), std::basic_string< _CharT, _Traits, _Alloc >::compare(), std::collate< _CharT >::do_compare(), std::collate< _CharT >::do_transform(), std::basic_string< char >::find(), std::basic_string< char >::find_first_not_of(), std::basic_string< char >::find_last_of(), std::match_results< _FwdIterT, _Alloc >::format(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::str(), and std::regex_traits< _Ch_type >::transform().

4.629.3.37 template<typename _CharT, typename _Traits, typename _Alloc> bool std::basic_string< _CharT, _Traits, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 811 of file basic_string.h.

Referenced by std::tr2::operator>>(), and std::operator>>().

4.629.3.38 template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 624 of file basic_string.h.

Referenced by std::basic_string< char >::crbegin(), std::basic_string< char >::rbegin(), std::regex_match(), std::regex_replace(), and std::regex_search().

4.629.3.39 template<typename _CharT, typename _Traits, typename _Alloc> const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 635 of file basic_string.h.

4.629.3.40 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::erase (size_type __pos = 0, size_type __n = npos) [inline]

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1362 of file basic_string.h.

Referenced by std::getline(), std::operator>>(), and std::basic_string< char >::pop_back().

4.629.3.41 template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (iterator __position) [inline]

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1378 of file basic_string.h.

4.629.3.42 template<typename _CharT , typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (iterator __first, iterator __last)

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 391 of file basic_string.tcc.

4.629.3.43 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find (const _CharT * *__s*, size_type *__pos*, size_type *__n*) const

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 737 of file basic_string.tcc.

References std::size().

Referenced by std::basic_string< char >::find(), and std::basic_string< char >::find_first_of().

4.629.3.44 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find (const basic_string< _CharT, _Traits, _Alloc > & *__str*, size_type *__pos* = 0) const [inline], [noexcept]

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1846 of file basic_string.h.

Referenced by std::basic_string< char >::find().

4.629.3.45 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find (const _CharT ∗ __s, size_type __pos = 0) const [inline]

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1861 of file basic_string.h.

4.629.3.46 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find (_CharT __c, size_type __pos = 0) const [noexcept]

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 760 of file basic_string.tcc.

References std::size().

4.629.3.47 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos = 0) const [inline], [noexcept]

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2079 of file basic_string.h.

Referenced by std::basic_string< char >::find_first_not_of().

4.629.3.48 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (const _CharT * __s, size_type __pos, size_type __n) const

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 852 of file basic_string.tcc.

References std::size().

4.629.3.49 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (const _CharT * __s, size_type __pos = 0) const [inline]

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2110 of file basic_string.h.

4.629.3.50 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (_CharT __c, size_type __pos = 0) const [noexcept]

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 864 of file basic_string.tcc.

References std::size().

4.629.3.51 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (const basic_string< _CharT, _Traits, _Alloc > & *__str*, size_type *__pos* = 0) const [inline], [noexcept]

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1952 of file basic_string.h.

Referenced by std::basic_string< char >::find_first_of().

4.629.3.52 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (const _CharT * *__s*, size_type *__pos*, size_type *__n*) const

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 816 of file basic_string.tcc.

References std::size().

4.629.3.53 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (const _CharT * *__s*, size_type *__pos* = 0) const [inline]

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 1982 of file basic_string.h.

4.629.3.54 template< typename _CharT, typename _Traits, typename _Alloc > size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (_CharT *__c*, size_type *__pos* = 0) const [inline], [noexcept]

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Note: equivalent to find(__c, __pos).

Definition at line 2001 of file basic_string.h.

4.629.3.55 template< typename _CharT, typename _Traits, typename _Alloc > size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (const basic_string< _CharT, _Traits, _Alloc > & *__str*, size_type *__pos* = npos) const [inline], [noexcept]

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2142 of file basic_string.h.

Referenced by std::basic_string< char >::find_last_not_of().

4.629.3.56 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (const _CharT * __s, size_type __pos, size_type __n) const

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 875 of file basic_string.tcc.

References std::size().

4.629.3.57 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (const _CharT * __s, size_type __pos = npos) const [inline]

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2173 of file basic_string.h.

4.629.3.58 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (_CharT __c, size_type __pos = npos) const [noexcept]

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 896 of file basic_string.tcc.

References std::size().

4.629.3.59 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (const basic_string< _CharT, _Traits, _Alloc > & *__str*, size_type *__pos* = npos) const [inline], [noexcept]

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2016 of file basic_string.h.

Referenced by std::basic_string< char >::find_last_of().

4.629.3.60 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (const _CharT * *__s*, size_type *__pos*, size_type *__n*) const

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 831 of file basic_string.tcc.

References std::size().

4.629.3.61 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (const _CharT * *__s*, size_type *__pos* = npos) const [inline]

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 2046 of file basic_string.h.

4.629.3.62 template< typename _CharT, typename _Traits, typename _Alloc> size_type **std::basic_string**< _CharT, _Traits, _Alloc >::find_last_of (_CharT *__c*, size_type *__pos* = **npos**) const [inline], [noexcept]

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Note: equivalent to rfind(__c, __pos).

Definition at line 2065 of file basic_string.h.

4.629.3.63 template< typename _CharT, typename _Traits, typename _Alloc> reference **std::basic_string**< _CharT, _Traits, _Alloc >::front ( ) [inline]

Returns a read/write reference to the data at the first element of the string.

Definition at line 897 of file basic_string.h.

4.629.3.64 template< typename _CharT, typename _Traits, typename _Alloc> const_reference **std::basic_string**< _CharT, _Traits, _Alloc >::front ( ) const [inline]

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 905 of file basic_string.h.

4.629.3.65 template< typename _CharT, typename _Traits, typename _Alloc> allocator_type **std::basic_string**< _CharT, _Traits, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Return copy of allocator used to construct this string.

Definition at line 1817 of file basic_string.h.

Referenced by std::basic_string< _CharT, _Traits, _Alloc >::assign(), std::basic_string< char >::basic_string(), std::basic_string< _CharT, _Traits, _Alloc >::swap(), and std::basic_string< char >::∼basic_string().

4.629.3.66 template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator *__p*, size_type *__n*, _CharT *__c*) `[inline]`

Insert multiple characters.

Parameters

__p	Iterator referencing location in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1185 of file basic_string.h.

Referenced by std::basic_string< char >::insert().

4.629.3.67 template<typename _CharT, typename _Traits, typename _Alloc> template<class _InputIterator > void std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator *__p*, _InputIterator *__beg*, _InputIterator *__end*) `[inline]`

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [__beg,__end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1202 of file basic_string.h.

4.629.3.68 template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator *__p*, initializer_list< _CharT > *__l*) `[inline]`

Insert an initializer_list of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Definition at line 1213 of file basic_string.h.

4.629.3.69 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (size_type __pos1, const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]

Insert value of a string.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1233 of file basic_string.h.

Referenced by std::basic_string< char >::insert().

4.629.3.70 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (size_type __pos1, const basic_string< _CharT, _Traits, _Alloc > & __str, size_type __pos2, size_type __n) [inline]

Insert a substring.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1255 of file basic_string.h.

Referenced by std::basic_string< char >::insert().

4.629.3.71 template<typename _CharT, typename _Traits , typename _Alloc > basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (size_type *__pos,* const _CharT * *__s,* size_type *__n*)

Insert a C substring.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 360 of file basic_string.tcc.

4.629.3.72 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (size_type *__pos,* const _CharT * *__s*) [inline]

Insert a C string.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1296 of file basic_string.h.

4.629.3.73 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (size_type __pos, size_type __n, _CharT __c) [inline]

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1319 of file basic_string.h.

4.629.3.74 template<typename _CharT, typename _Traits, typename _Alloc> iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (iterator __p, _CharT __c) [inline]

Insert one character.

Parameters

__p	Iterator referencing position in string to insert at.
__c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1337 of file basic_string.h.

4.629.3.75 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::length ( ) const [inline], [noexcept]

Returns the number of characters in the string, not including any null-termination.

Definition at line 721 of file basic_string.h.

Referenced by std::collate< _CharT >::do_compare(), and std::collate< _CharT >::do_transform().

4.629.3.76 template<typename _CharT, typename _Traits, typename _Alloc> size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the size() of the largest possible string.

Definition at line 726 of file basic_string.h.

Referenced by std::getline(), and std::operator>>().

4.629.3.77 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+= (const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Definition at line 932 of file basic_string.h.

4.629.3.78 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+= (const _CharT * __s) [inline]

Append a C string.

Parameters

__s	The C string to append.

Returns

Reference to this string.

Definition at line 941 of file basic_string.h.

4.629.3.79 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+= (_CharT __c) [inline]

Append a character.

Parameters

__c	The character to append.

Returns

Reference to this string.

Definition at line 950 of file basic_string.h.

4.629.3.80 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator+= (initializer_list< _CharT > __l) [inline]

Append an initializer_list of characters.

Parameters

__l	The initializer_list of characters to be appended.

Returns

Reference to this string.

Definition at line 963 of file basic_string.h.

4.629.3.81 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]

Assign the value of *str* to this string.

Parameters

__str	Source string.

Definition at line 546 of file basic_string.h.

4.629.3.82 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (const _CharT * __s) [inline]

Copy contents of *s* into this string.

Parameters

__s	Source null-terminated string.

Definition at line 554 of file basic_string.h.

4.629.3.83 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (_CharT __c) [inline]

Set value to string of length 1.

Parameters

__c	Source character.

Assigning to a character makes this string length 1 and (∗this)[0] == *c*.

Definition at line 565 of file basic_string.h.

4.629.3.84 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (basic_string< _CharT, _Traits, _Alloc > && __str) [inline]

Move assign the value of *str* to this string.

Parameters

__str	Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 580 of file basic_string.h.

4.629.3.85 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (initializer_list< _CharT > *__l*) [inline]

Set value to string constructed from initializer list.

Parameters

__l	std::initializer_list.

Definition at line 592 of file basic_string.h.

4.629.3.86 template<typename _CharT, typename _Traits, typename _Alloc> const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (size_type *__pos*) const [inline]

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 826 of file basic_string.h.

Referenced by std::basic_string< char >::back(), and std::basic_string< char >::front().

4.629.3.87 template<typename _CharT, typename _Traits, typename _Alloc> reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (size_type *__pos*) [inline]

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().) Unshares the string.

Definition at line 843 of file basic_string.h.

4.629.3.88 template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::pop_back ( ) [inline]

Remove the last character.

The string must be non-empty.

Definition at line 1407 of file basic_string.h.

4.629.3.89 template<typename _CharT, typename _Traits, typename _Alloc> void std::basic_string< _CharT, _Traits, _Alloc >::push_back (_CharT *__c*) [inline]

Append a single character.

Parameters

__c	Character to append.

Definition at line 1052 of file basic_string.h.

Referenced by std::collate< _CharT >::do_transform(), std::basic_string< char >::operator+=(), std::tr2::operator>>(), std::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

4.629.3.90 template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 644 of file basic_string.h.

4.629.3.91 template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 653 of file basic_string.h.

4.629.3.92 template<typename _CharT, typename _Traits, typename _Alloc> reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 662 of file basic_string.h.

4.629.3.93 template<typename _CharT, typename _Traits, typename _Alloc> const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 671 of file basic_string.h.

4.629.3.94 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (size_type *__pos*, size_type *__n*, const basic_string< _CharT, _Traits, _Alloc > & *__str*) [inline]

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1429 of file basic_string.h.

Referenced by std::basic_string< char >::append(), std::basic_string< char >::assign(), std::basic_string< char >::insert(), and std::basic_string< char >::replace().

4.629.3.95 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (size_type *__pos1,* size_type *__n1,* const basic_string< _CharT, _Traits, _Alloc > & *__str,* size_type *__pos2,* size_type *__n2*) [inline]

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1451 of file basic_string.h.

Referenced by std::basic_string< char >::replace().

4.629.3.96 template<typename _CharT, typename _Traits , typename _Alloc > **basic_string**< _CharT, _Traits, _Alloc > & **std::basic_string**< _CharT, _Traits, _Alloc >::replace (size_type *__pos,* size_type *__n1,* const _CharT * *__s,* size_type *__n2*)

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first __*n2* characters of __*s* are inserted, or all of __*s* if __*n2* is too large. If __*pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 414 of file basic_string.tcc.

4.629.3.97 template<typename _CharT, typename _Traits, typename _Alloc> **basic_string& std::basic_string**< _CharT, _Traits, _Alloc >::replace (size_type *__pos,* size_type *__n1,* const _CharT * *__s*) `[inline]`

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of __*s* are inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1496 of file basic_string.h.

4.629.3.98 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (size_type *__pos,* size_type *__n1,* size_type *__n2,* _CharT *__c*) [inline]

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1520 of file basic_string.h.

4.629.3.99 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (iterator *__i1,* iterator *__i2,* const basic_string< _CharT, _Traits, _Alloc > & *__str*) [inline]

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1538 of file basic_string.h.

Referenced by std::basic_string< char >::replace().

4.629.3.100 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (iterator *__i1,* iterator *__i2,* const _CharT * *__s,* size_type *__n*) [inline]

Replace range of characters with C substring.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.
__n	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1557 of file basic_string.h.

4.629.3.101 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (iterator *__i1,* iterator *__i2,* const _CharT * *__s*) [inline]

Replace range of characters with C string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1578 of file basic_string.h.

4.629.3.102 template<typename _CharT, typename _Traits, typename _Alloc> basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (iterator *__i1,* iterator *__i2,* size_type *__n,* _CharT *__c*) [inline]

Replace range of characters with multiple characters.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__n	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, *__n* copies of *__c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1599 of file basic_string.h.

4.629.3.103 template< typename _CharT, typename _Traits, typename _Alloc > template< class _InputIterator > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (iterator *__i1*, iterator *__i2*, _InputIterator *__k1*, _InputIterator *__k2*) `[inline]`

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1623 of file basic_string.h.

4.629.3.104 template< typename _CharT, typename _Traits, typename _Alloc > basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (iterator *__i1*, iterator *__i2*, initializer_list< _CharT > *__l*) `[inline]`

Replace range of characters with initializer_list.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1692 of file basic_string.h.

Referenced by std::basic_string< char >::replace().

4.629.3.105 template<typename _CharT , typename _Traits , typename _Alloc > void **std::basic_string**< _CharT, _Traits, _Alloc >::reserve (size_type *__res_arg* = `0`)

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 502 of file basic_string.tcc.

References std::size().

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::tr2::operator>>(), std::operator>>(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_string< char >::push_back(), and std::basic_string< char >::shrink_to_fit().

4.629.3.106 template<typename _CharT, typename _Traits , typename _Alloc > void **std::basic_string**< _CharT, _Traits, _Alloc >::resize (size_type *__n*, _CharT *__c*)

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.
__c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to *__c*.

Definition at line 640 of file basic_string.tcc.

References std::size().

Referenced by std::money_get< _CharT, _InIter >::do_get().

4.629.3.107 template<typename _CharT, typename _Traits, typename _Alloc> void **std::basic_string**< _CharT, _Traits, _Alloc >::resize (size_type *__n*) `[inline]`

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 753 of file basic_string.h.

Referenced by std::basic_string< char >::resize().

4.629.3.108 template<typename _CharT, typename _Traits, typename _Alloc> size_type **std::basic_string**< _CharT, _Traits, _Alloc >::rfind (const **basic_string**< _CharT, _Traits, _Alloc > & *__str,* size_type *__pos* = **npos**) const `[inline]`, `[noexcept]`

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1891 of file basic_string.h.

Referenced by std::basic_string< char >::find_last_of(), and std::basic_string< char >::rfind().

4.629.3.109 template<typename _CharT, typename _Traits , typename _Alloc > **basic_string**< _CharT, _Traits, _Alloc >::size_type **std::basic_string**< _CharT, _Traits, _Alloc >::rfind (const _CharT ∗ *__s,* size_type *__pos,* size_type *__n*) const

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 778 of file basic_string.tcc.

References std::min(), and std::size().

4.629.3.110 template<typename _CharT, typename _Traits, typename _Alloc> size_type **std::basic_string**< _CharT, _Traits, _Alloc >::rfind (const _CharT * *__s*, size_type *__pos* = **npos**) const [inline]

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

Definition at line 1921 of file basic_string.h.

4.629.3.111 template<typename _CharT, typename _Traits , typename _Alloc > **basic_string**< _CharT, _Traits, _Alloc >::size_type **std::basic_string**< _CharT, _Traits, _Alloc >::rfind (_CharT *__c*, size_type *__pos* = **npos**) const [noexcept]

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

Definition at line 799 of file basic_string.tcc.

References std::size().

4.629.3.112 template<typename _CharT, typename _Traits, typename _Alloc> void **std::basic_string**< _CharT, _Traits, _Alloc >::shrink_to_fit ( ) [inline]

A non-binding request to reduce capacity() to size().

Definition at line 759 of file basic_string.h.

4.629.3.113 template<typename _CharT, typename _Traits, typename _Alloc> size_type **std::basic_string**< _CharT, _Traits, _Alloc >::size ( ) const [inline], [noexcept]

Returns the number of characters in the string, not including any null-termination.

Definition at line 715 of file basic_string.h.

Referenced by std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< char >::assign(), std::basic_string< char >::at(), std::basic_string< char >::back(), std::basic_string< char >::cend(), std::basic_string< char

>::clear(), std::basic_string< char >::compare(), std::basic_string< _CharT, _Traits, _Alloc >::compare(), std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset(), std::basic_string< char >::empty(), std::basic_string< char >::end(), std::basic_string< char >::find(), std::basic_string< char >::find_first_not_of(), std::basic_string< char >::find_last_of(), std::match_results< _FwdIterT, _Alloc >::format(), std::basic_string< char >::insert(), std::operator+(), std::tr2::operator>>(), std::basic_string< char >::operator[](), std::basic_string< char >::pop_back(), std::basic_string< char >::push_back(), std::basic_string< char >::replace(), std::basic_string< char >::shrink_to_fit(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::str(), and std::regex_traits< _Ch_type >::transform().

4.629.3.114 template<typename _CharT, typename _Traits, typename _Alloc> basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (size_type __pos = 0, size_type __n = npos) const [inline]

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

Definition at line 2206 of file basic_string.h.

4.629.3.115 template<typename _CharT , typename _Traits , typename _Alloc > void std::basic_string< _CharT, _Traits, _Alloc >::swap (basic_string< _CharT, _Traits, _Alloc > & __s)

Swap contents with another string.

Parameters

__s	String to swap with.

Exchanges the contents of this string with that of *__s* in constant time.

Definition at line 519 of file basic_string.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::get_allocator().

Referenced by std::basic_string< char >::assign(), std::basic_string< char >::operator=(), and std::swap().

4.629.4 Member Data Documentation

4.629.4.1 template<typename _CharT, typename _Traits, typename _Alloc> const basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc >::npos [static]

Value returned by various member functions when they fail.

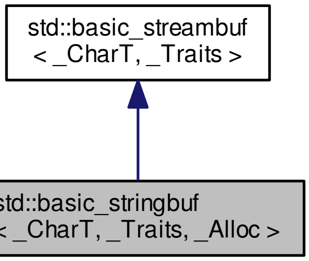
Definition at line 285 of file basic_string.h.

The documentation for this class was generated from the following files:

- basic_string.h
- basic_string.tcc

4.630 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

Inheritance diagram for std::basic_stringbuf< _CharT, _Traits, _Alloc >:



Public Types

- typedef __string_type::size_type **__size_type**
- typedef basic_streambuf
  < char_type, traits_type > **__streambuf_type**
- typedef basic_string
  < char_type, _Traits, _Alloc > **__string_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- basic_stringbuf (ios_base::openmode __mode=ios_base::in|ios_base::out)
- basic_stringbuf (const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out)
- locale getloc () const
- streamsize in_avail ()
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type $*$__s, streamsize __n)
- int_type snextc ()

- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type ∗__s, streamsize __n)
- __string_type str () const
- void str (const __string_type &__s)
- int_type sungetc ()

- basic_streambuf ∗ pubsetbuf (char_type ∗__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_pbump** (char_type ∗__pbeg, char_type ∗__pend, off_type __off)
- void **_M_stringbuf_init** (ios_base::openmode __mode)
- void **_M_sync** (char_type ∗__base, __size_type __i, __size_type __o)
- void **_M_update_egptr** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type ∗ setbuf (char_type ∗__s, streamsize __n)
- void setg (char_type ∗__gbeg, char_type ∗__gnext, char_type ∗__gend)
- void setp (char_type ∗__pbeg, char_type ∗__pend)
- virtual streamsize showmanyc ()
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type ∗__s, streamsize __n)
- virtual streamsize xsputn (const char_type ∗__s, streamsize __n)

- char_type ∗ eback () const
- char_type ∗ gptr () const
- char_type ∗ egptr () const

- char_type ∗ pbase () const
- char_type ∗ pptr () const
- char_type ∗ epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type ∗ _M_in_beg
- char_type ∗ _M_in_cur
- char_type ∗ _M_in_end
- ios_base::openmode _M_mode
- char_type ∗ _M_out_beg
- char_type ∗ _M_out_cur
- char_type ∗ _M_out_end
- __string_type **_M_string**

4.630.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>class std::basic_stringbuf< _CharT, _Traits, _Alloc >

The actual work of input and output (for std::string).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.)

For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 64 of file sstream.

4.630.2 Constructor & Destructor Documentation

4.630.2.1 template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (ios_base::openmode *__mode* = ios_base::in | ios_base::out) `[inline]`, `[explicit]`

Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

Definition at line 98 of file sstream.

4.630.2.2 template<typename _CharT, typename _Traits, typename _Alloc> std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (const __string_type & *__str*, ios_base::openmode *__mode* = ios_base::in | ios_base::out) `[inline]`, `[explicit]`

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

Definition at line 111 of file sstream.

4.630.3 Member Function Documentation

4.630.3.1 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::eback ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 482 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sungetc().

4.630.3.2 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::egptr ( ) const [inline], [protected], [inherited]

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 488 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.630.3.3 template<typename _CharT, typename _Traits> char_type* std::basic_streambuf< _CharT, _Traits >::epptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence

- epptr() returns the end pointer for the output sequence

Definition at line 535 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.630.3.4 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::gbump (int *__n*) `[inline]`, `[protected]`, `[inherited]`

Moving the read position.

Parameters

__n	The delta by which to move.

This just advances the read position without returning any data.

Definition at line 498 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.630.3.5 template<typename _CharT, typename _Traits> locale **std::basic_streambuf**< _CharT, _Traits >::getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 226 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.630.3.6 template<typename _CharT, typename _Traits> char_type* **std::basic_streambuf**< _CharT, _Traits >::gptr ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 485 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

4.630.3.7 template<typename _CharT, typename _Traits> virtual void **std::basic_streambuf**< _CharT, _Traits >::imbue (const **locale** & *__loc*) [inline], [protected], [virtual], [inherited]

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 576 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

4.630.3.8 template<typename _CharT, typename _Traits> **streamsize std::basic_streambuf**< _CharT, _Traits >::in_avail ( ) [inline], [inherited]

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 284 of file streambuf.

4.630.3.9 template<class _CharT , class _Traits , class _Alloc > **basic_stringbuf**< _CharT, _Traits, _Alloc >::int_type **basic_stringbuf**< _CharT, _Traits, _Alloc >::overflow (int_type *__c* = `traits_type::eof()`) [protected], [virtual]

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __*c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __*c* is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 80 of file sstream.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::assign(), std::max(), std::min(), std::ios_base::out, std::basic_string< _CharT, _Traits, _Alloc >::push_back(), and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

4.630.3.10 template<class _CharT , class _Traits , class _Alloc > **basic_stringbuf**< _CharT, _Traits, _Alloc >::int_type **basic_stringbuf**< _CharT, _Traits, _Alloc >::pbackfail (int_type *__c* = `traits_type::eof()`) `[protected]`, `[virtual]`

Tries to back up the input sequence.

Parameters

__c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 46 of file sstream.tcc.

References std::ios_base::out.

4.630.3.11 template<typename _CharT, typename _Traits> char_type* **std::basic_streambuf**< _CharT, _Traits >::pbase ( ) const `[inline]`, `[protected]`, `[inherited]`

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 529 of file streambuf.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.630.3.12 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::pbump (int *__n*) [inline], [protected], [inherited]

Moving the write position.

Parameters

__n	The delta by which to move.

This just advances the write position without returning any data.

Definition at line 545 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc().

4.630.3.13 template<typename _CharT, typename _Traits> char_type* **std::basic_streambuf**< _CharT, _Traits >::pptr ( ) const [inline], [protected], [inherited]

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 532 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sputc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.630.3.14 template<typename _CharT, typename _Traits> **locale std::basic_streambuf**< _CharT, _Traits >::pubimbue (const **locale** & *__loc*) [inline], [inherited]

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

Definition at line 209 of file streambuf.

4.630.3.15 template<typename _CharT, typename _Traits> pos_type **std::basic_streambuf**< _CharT, _Traits >::pubseekoff (**off_type** *__off*, **ios_base::seekdir** *__way*, **ios_base::openmode** *__mode* = **ios_base::in** \| **ios_base::out**) [inline], [inherited]

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

Definition at line 251 of file streambuf.

4.630.3.16 template<typename _CharT, typename _Traits> pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (pos_type *__sp*, ios_base::openmode *__mode* = ios_base::in | ios_base::out) [inline], [inherited]

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

Definition at line 263 of file streambuf.

4.630.3.17 template<typename _CharT, typename _Traits> basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (char_type * *__s*, streamsize *__n*) [inline], [inherited]

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 239 of file streambuf.

4.630.3.18 template<typename _CharT, typename _Traits> int std::basic_streambuf< _CharT, _Traits >::pubsync ( ) [inline], [inherited]

Calls virtual sync function.

Definition at line 271 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::sync().

4.630.3.19 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ( ) [inline], [inherited]

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 316 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::istreambuf_iterator< _CharT, _Traits >::operator++(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.630.3.20 template<class _CharT , class _Traits , class _Alloc > basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff (off_type , ios_base::seekdir , ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 149 of file sstream.tcc.

References std::ios_base::cur, std::ios_base::end, std::ios_base::in, and std::ios_base::out.

4.630.3.21 template<class _CharT , class _Traits , class _Alloc > basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos (pos_type , ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]

Alters the stream positions.

Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 197 of file sstream.tcc.

References std::ios_base::in, and std::ios_base::out.

4.630.3.22 template<typename _CharT, typename _Traits, typename _Alloc> virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (char_type * __s, streamsize __n) [inline], [protected], [virtual]

Manipulates the buffer.

Parameters

__s	Pointer to a buffer area.
__n	Size of *__s*.

Returns

`this`

If no buffer has already been created, and both *__s* and *__n* are non-zero, then `__s` is used as a buffer; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25s02.html` for more.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 202 of file sstream.

References std::basic_string< _CharT, _Traits, _Alloc >::clear().

4.630.3.23 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setg (**char_type** ∗ *__gbeg,* **char_type** ∗ *__gnext,* **char_type** ∗ *__gend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

> *__gbeg* == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Definition at line 509 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), and std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.630.3.24 template<typename _CharT, typename _Traits> void **std::basic_streambuf**< _CharT, _Traits >::setp (**char_type** ∗ *__pbeg,* **char_type** ∗ *__pend*) `[inline]`, `[protected]`, `[inherited]`

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

> *__pbeg* == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Definition at line 555 of file streambuf.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer().

4.630.3.25 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sgetc ( ) `[inline]`, `[inherited]`

Getting the next character.

Returns

> The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 338 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::basic_streambuf< char_type, traits_type >::snextc().

4.630.3.26 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (char_type * __s, streamsize __n) [inline], [inherited]

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill *__s*[0] through *__s*[__n-1] with characters from the input sequence, if possible.

Definition at line 357 of file streambuf.

4.630.3.27 template<typename _CharT, typename _Traits, typename _Alloc> virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc ( ) [inline], [protected], [virtual]

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.
The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*
The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 170 of file sstream.

References std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), and std::ios_base::in.

4.630.3.28 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::snextc ( ) [inline], [inherited]

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 298 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.630.3.29 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (char_type __c) [inline], [inherited]

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.

Definition at line 372 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::putback().

4.630.3.30 template<typename _CharT, typename _Traits> int_type std::basic_streambuf< _CharT, _Traits >::sputc (char_type __c) [inline], [inherited]

Entry point for all single-character output functions.

Parameters

__c	A character to output.

Returns

__c, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores *__c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 424 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::get(), and std::ostreambuf_iterator< _CharT, _Traits >::operator=().

4.630.3.31 template<typename _CharT, typename _Traits> streamsize std::basic_streambuf< _CharT, _Traits >::sputn (const char_type * __s, streamsize __n) [inline], [inherited]

Entry point for all single-character output functions.

Parameters

__s	A buffer read area.
__n	A count.

One of two public output functions.

Returns xsputn(__s,__n). The effect is to write *__s*[0] through *__s*[__n-1] to the output sequence, if possible.

Definition at line 450 of file streambuf.

4.630.3.32 template<typename _CharT, typename _Traits, typename _Alloc> **__string_type std::basic_stringbuf**< _CharT, _Traits, _Alloc >::str ( ) const `[inline]`

Copying out the string buffer.

Returns

A copy of one of the underlying sequences.

If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1

Definition at line 126 of file sstream.

References std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().

4.630.3.33 template<typename _CharT, typename _Traits, typename _Alloc> void **std::basic_stringbuf**< _CharT, _Traits, _Alloc >::str (const **__string_type** & *__s*) `[inline]`

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Deallocates any previous stored sequence, then copies *s* to use as a new one.

Definition at line 150 of file sstream.

References std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode, std::basic_string< _CharT, _Traits, _Alloc >::assign(), std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

4.630.3.34 template<typename _CharT, typename _Traits> int_type **std::basic_streambuf**< _CharT, _Traits >::sungetc ( ) `[inline]`, `[inherited]`

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 397 of file streambuf.

Referenced by std::basic_istream< _CharT, _Traits >::unget().

4.630.3.35 template<typename _CharT, typename _Traits> virtual int **std::basic_streambuf**< _CharT, _Traits >::sync (void) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char_type, traits_type >, and __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 627 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pubsync().

4.630.3.36 template<typename _CharT, typename _Traits> virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ( ) `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.

Definition at line 700 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc().

4.630.3.37 template<class _CharT , class _Traits , class _Alloc > basic_stringbuf< _CharT, _Traits, _Alloc >::int_type basic_stringbuf< _CharT, _Traits, _Alloc >::underflow ( ) `[protected]`, `[virtual]`

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch25.html`

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

Definition at line 131 of file sstream.tcc.

References std::ios_base::in.

4.630.3.38 template<typename _CharT , typename _Traits > streamsize basic_streambuf< _CharT, _Traits >::xsgetn (char_type ∗ *__s*, streamsize *__n*) `[protected]`, `[virtual]`, `[inherited]`

Multiple character extraction.

Parameters

__s	A buffer area.
__n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 46 of file streambuf.tcc.

References std::min().

Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

4.630.3.39 template<typename _CharT , typename _Traits > **streamsize basic_streambuf< _CharT, _Traits >::xsputn (const char_type * *__s*, streamsize *__n*)** `[protected]`, `[virtual]`, `[inherited]`

Multiple character insertion.

Parameters

__s	A buffer area.
__n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, and std::basic_filebuf< char_type, traits_type >.

Definition at line 80 of file streambuf.tcc.

References std::min().

Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

4.630.4 Member Data Documentation

4.630.4.1 template<typename _CharT, typename _Traits> **locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale** `[protected]`, `[inherited]`

Current locale setting.

Definition at line 192 of file streambuf.

Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf(), std::basic_streambuf< char_type, traits_type >::getloc(), and std::basic_streambuf< char_type, traits_type >::pubimbue().

4.630.4.2 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::_M_in_beg `[protected], [inherited]`

Start of get area.

Definition at line 184 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::eback(), and std::basic_streambuf< char_type, traits_type >::setg().

4.630.4.3 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::_M_in_cur `[protected], [inherited]`

Current read area.

Definition at line 185 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::gbump(), std::basic_streambuf< char_type, traits_type >::gptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.630.4.4 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::_M_in_end `[protected], [inherited]`

End of get area.

Definition at line 186 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::egptr(), and std::basic_streambuf< char_type, traits_type >::setg().

4.630.4.5 template<typename _CharT, typename _Traits, typename _Alloc> **ios_base::openmode std::basic_stringbuf**< _CharT, _Traits, _Alloc >::_M_mode `[protected]`

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 83 of file sstream.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::str().

4.630.4.6 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::_M_out_beg `[protected], [inherited]`

Start of put area.

Definition at line 187 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbase(), and std::basic_streambuf< char_type, traits_type >::setp().

4.630.4.7 template<typename _CharT, typename _Traits> char_type∗ **std::basic_streambuf**< _CharT, _Traits >::_M_out_cur `[protected], [inherited]`

Current put area.

Definition at line 188 of file streambuf.

Referenced by std::basic_streambuf< char_type, traits_type >::pbump(), std::basic_streambuf< char_type, traits_type >::pptr(), and std::basic_streambuf< char_type, traits_type >::setp().

4.630.4.8 template<typename _CharT, typename _Traits> char_type∗ std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]

End of put area.

Definition at line 189 of file streambuf.

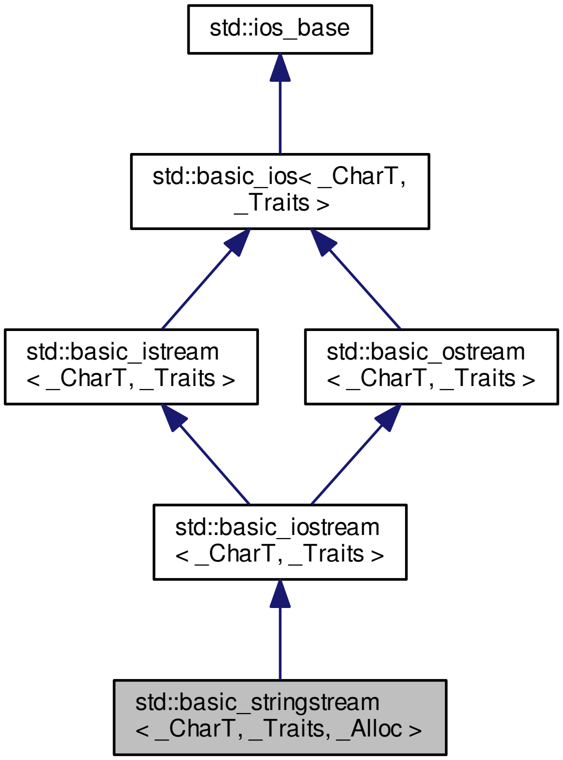
Referenced by std::basic_streambuf< char_type, traits_type >::epptr(), and std::basic_streambuf< char_type, traits_type >::setp().

The documentation for this class was generated from the following files:

- sstream
- sstream.tcc

4.631 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference

Inheritance diagram for std::basic_stringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_iostream < char_type, traits_type > **__iostream_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf < _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf < _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_stringstream (ios_base::openmode __m=ios_base::out|ios_base::in)
- basic_stringstream (const __string_type &__str, ios_base::openmode __m=ios_base::out|ios_base::in)
- ∼basic_stringstream ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (const void ∗__p)
- __ostream_type & operator<< (__streambuf_type ∗__sb)
- __istream_type & operator>> (void ∗&__p)
- __istream_type & operator>> (__streambuf_type ∗__sb)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- basic_streambuf< _CharT,
  _Traits > ∗ rdbuf (basic_streambuf< _CharT, _Traits > ∗__sb)
- __stringbuf_type ∗ rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (pos_type)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type str () const
- void str (const __string_type &__s)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > ∗ tie () const
- basic_ostream< _CharT, _Traits > ∗ tie (basic_ostream< _CharT, _Traits > ∗__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(∗__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(∗__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(∗__pf)(ios_base &))

Extractors

All the `operator>>` *functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type ∗__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type ∗__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type ∗__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type ∗__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type ∗__s, streamsize __n)
- streamsize readsome (char_type ∗__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)

  - __istream_type & seekg (off_type, ios_base::seekdir)

- operator void ∗ () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(∗__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(∗__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(∗__pf)(ios_base &))

Inserters

All the `operator<<` *functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.*

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

  - __ostream_type & operator<< (long __n)
  - __ostream_type & operator<< (unsigned long __n)
  - __ostream_type & operator<< (bool __n)
  - __ostream_type & operator<< (short __n)
  - __ostream_type & operator<< (unsigned short __n)
  - __ostream_type & operator<< (int __n)
  - __ostream_type & operator<< (unsigned int __n)
  - __ostream_type & operator<< (long long __n)
  - __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

  - __ostream_type & put (char_type __c)
  - void _M_write (const char_type ∗__s, streamsize __n)
  - __ostream_type & write (const char_type ∗__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT >
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT >
  __ostream_type & **_M_insert** (_ValueT __v)
- void init (basic_streambuf< _CharT, _Traits > *__sb)

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- const __ctype_type ∗ **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type ∗ **_M_num_get**
- const __num_put_type ∗ **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > ∗ **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > ∗ **_M_tie**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.631.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>class std::basic_stringstream< _CharT, _Traits, _Alloc >

Controlling input and output for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports reading from and writing to objects of type std::basic_string, using the inherited functions from std::basic_iostream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

Definition at line 502 of file sstream.

4.631.2 Member Typedef Documentation

4.631.2.1 template<typename _CharT, typename _Traits> typedef **num_put**<_CharT, **ostreambuf_iterator**<_CharT, _Traits> > **std::basic_ios**< _CharT, _Traits >::**__num_put_type** `[inherited]`

These are non-standard types.

Definition at line 88 of file basic_ios.h.

4.631.2.2 typedef void(∗ std::ios_base::event_callback)(**event** __e, **ios_base** &__b, int __i) `[inherited]`

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.631.2.3 typedef _Ios_Fmtflags std::ios_base::fmtflags `[inherited]`

This is a bitmask type.

`_Ios_Fmtflags` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.631.2.4 typedef _Ios_Iostate std::ios_base::iostate `[inherited]`

This is a bitmask type.

`_Ios_Iostate` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.631.2.5 typedef _Ios_Openmode std::ios_base::openmode `[inherited]`

This is a bitmask type.

`_Ios_Openmode` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.631.2.6 typedef _Ios_Seekdir std::ios_base::seekdir `[inherited]`

This is an enumerated type.

`_Ios_Seekdir` `is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.631.3 Member Enumeration Documentation

4.631.3.1 enum std::ios_base::event `[inherited]`

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.631.4 Constructor & Destructor Documentation

4.631.4.1 template<typename _CharT , typename _Traits , typename _Alloc > **std::basic_stringstream**< _CharT, _Traits, _Alloc >::**basic_stringstream (ios_base::openmode** *__m* = **ios_base::out | ios_base::in)** `[inline]`, `[explicit]`

Default constructor starts with an empty string buffer.

Parameters

__m	Whether the buffer can read, or write, or both.

Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 537 of file sstream.

References std::basic_ios< _CharT, _Traits >::init().

4.631.4.2 template<typename _CharT , typename _Traits , typename _Alloc > **std::basic_stringstream**< _CharT, _Traits, _Alloc >::**basic_stringstream (const __string_type &** *__str*, **ios_base::openmode** *__m* = **ios_base::out | ios_base::in)** `[inline]`, `[explicit]`

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__m	Whether the buffer can read, or write, or both.

Initializes `sb` using *__str* and `__m`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 553 of file sstream.

References std::basic_ios< _CharT, _Traits >::init().

4.631.4.3 template<typename _CharT , typename _Traits , typename _Alloc > **std::basic_stringstream**< _CharT, _Traits, _Alloc >::~**basic_stringstream ( )** `[inline]`

The destructor does nothing.

The buffer is deallocated by the stringbuf object, not the formatting stream.

Definition at line 564 of file sstream.

4.631.5 Member Function Documentation

4.631.5.1 const **locale**& std::ios_base::_M_getloc ( ) const `[inline]`, `[inherited]`

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.631.5.2 template<typename _CharT, typename _Traits> void **std::basic_ostream**< _CharT, _Traits >::_M_write (const char_type * *__s*, **streamsize** *__n*) [inline], [inherited]

Core write functionality, without sentry.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Definition at line 311 of file ostream.

Referenced by std::basic_ostream< _CharT, _Traits >::write().

4.631.5.3 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::bad ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 205 of file basic_ios.h.

4.631.5.4 template<typename _CharT , typename _Traits > void **basic_ios**< _CharT, _Traits >::clear (**iostate** *__state* = **goodbit**) [inherited]

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic_ios.tcc.

Referenced by std::basic_ios< char, _Traits >::exceptions(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.631.5.5 template<typename _CharT , typename _Traits > **basic_ios**< _CharT, _Traits > & **basic_ios**< _CharT, _Traits >::copyfmt (const **basic_ios**< _CharT, _Traits > & *__rhs*) [inherited]

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

Definition at line 63 of file basic_ios.tcc.

References std::basic_ios< _CharT, _Traits >::exceptions(), std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), std::basic_ios< _CharT, _Traits >::tie(), std::tie(), and std::ios_base::width().

4.631.5.6 template<typename _CharT, typename _Traits> bool std::basic_ios< _CharT, _Traits >::eof ( ) const [inline], [inherited]

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 184 of file basic_ios.h.

4.631.5.7 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::exceptions ( ) const [inline], [inherited]

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 216 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

4.631.5.8 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::exceptions (iostate *__except*) [inline], [inherited]

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (
      __gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr << "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr << "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

Definition at line 251 of file basic_ios.h.

4.631.5.9 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::fail ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 195 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::operator void ∗(), std::basic_ios< char, _Traits >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::regex_traits< _Ch_type >::value().

4.631.5.10 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill ( ) const `[inline]`, `[inherited]`

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 364 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::basic_ios< char, _Traits >::fill().

4.631.5.11 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::fill (**char_type** *__ch*) `[inline]`, `[inherited]`

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

Definition at line 384 of file basic_ios.h.

4.631.5.12 fmtflags std::ios_base::flags ( ) const [inline], [inherited]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.631.5.13 fmtflags std::ios_base::flags (fmtflags *__fmtfl*) [inline], [inherited]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.631.5.14 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::flush ( ) [inherited]

Synchronizing the stream buffer.

Returns

∗this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::flush().

4.631.5.15 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::gcount ( ) const [inline], [inherited]

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

4.631.5.16 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::get (void) [inherited]

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 236 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.17 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type & __c) [inherited]

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

∗this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 272 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.18 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (char_type ∗ __s, streamsize __n, char_type __delim) [inherited]

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 309 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.631.5.19 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (char_type ∗ *__s*, streamsize *__n*) `[inline]`, `[inherited]`

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

Definition at line 354 of file istream.

4.631.5.20 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*, char_type *__delim*) `[inherited]`

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 356 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), std::basic_streambuf< _CharT, _Traits >::snextc(), and std::basic_streambuf< _CharT, _Traits >::sputc().

4.631.5.21 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::get (__streambuf_type & *__sb*) [inline], [inherited]

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

∗this

Returns `get`(__sb,widen('\n')).

Definition at line 387 of file istream.

4.631.5.22 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::getline (char_type ∗ *__s*, streamsize *__n*, char_type *__delim*) [inherited]

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

∗this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 400 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.631.5.23 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::getline (char_type ∗ *__s*, streamsize *__n*) [inline], [inherited]

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.

Returns

∗this

Returns `getline`(__s,__n,widen('\n')).

Definition at line 427 of file istream.

Referenced by std::basic_istream< char >::getline().

4.631.5.24 locale std::ios_base::getloc ( ) const [inline], [inherited]

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.631.5.25 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::good ( ) const `[inline]`, `[inherited]`

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Definition at line 174 of file basic_ios.h.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.631.5.26 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (**streamsize** *__n*, int_type *__delim*) `[inherited]`

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

∗this

Extracts characters and throws them away until one of the following happens:

- if *__n* `!= std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 555 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.631.5.27 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (**streamsize** *__n*) `[inherited]`

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 493 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

4.631.5.28 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::ignore (void) `[inherited]`

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 460 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.29 template<typename _CharT , typename _Traits > **locale basic_ios**< _CharT, _Traits >::imbue (const locale & *__loc*) `[inherited]`

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 114 of file basic_ios.tcc.

References std::ios_base::imbue().

Referenced by std::operator<<().

4.631.5.30 template<typename _CharT, typename _Traits> void **basic_ios**< _CharT, _Traits >::init (**basic_streambuf**< _CharT, _Traits > * *__sb*) `[protected]`, `[inherited]`

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic_ios.tcc.

Referenced by std::basic_fstream< _CharT, _Traits >::basic_fstream(), std::basic_ifstream< _CharT, _Traits >::basic_ifstream(), std::basic_ios< char, _Traits >::basic_ios(), std::basic_istream< char >::basic_istream(), std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream(), std::basic_ofstream< _CharT, _Traits >::basic_ofstream(), std::basic_ostream< char >::basic_ostream(), std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream(), and std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream().

4.631.5.31 long& std::ios_base::iword (int *__ix*) [inline], [inherited]

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.631.5.32 template<typename _CharT, typename _Traits> char std::basic_ios< _CharT, _Traits >::narrow (char_type *__c*, char *__dfault*) const [inline], [inherited]

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 424 of file basic_ios.h.

4.631.5.33 template<typename _CharT, typename _Traits> std::basic_ios< _CharT, _Traits >::operator void * ( ) const [inline], [inherited]

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 115 of file basic_ios.h.

4.631.5.34 template<typename _CharT, typename _Traits> bool **std::basic_ios**< _CharT, _Traits >::operator! ( ) const `[inline]`, `[inherited]`

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 119 of file basic_ios.h.

4.631.5.35 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ostream_type &(*)(__ostream_type &)** *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 108 of file ostream.

4.631.5.36 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**__ios_type &(*)(__ios_type &)** *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 117 of file ostream.

4.631.5.37 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (**ios_base &(*)(ios_base &)** *__pf*) `[inline]`, `[inherited]`

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

Definition at line 127 of file ostream.

4.631.5.38 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (long *__n*) `[inline]`, `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

4.631.5.39 template<typename _CharT, typename _Traits> **__ostream_type& std::basic_ostream**< _CharT, _Traits >::operator<< (unsigned long *__n*) `[inline]`, `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file ostream.

4.631.5.40 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (bool __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

4.631.5.41 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (short __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.631.5.42 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned short __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

4.631.5.43 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (int __n) [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file ostream.tcc.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, and std::ios_base::oct.

4.631.5.44 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned int __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

4.631.5.45 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long long __n) [inline], [inherited]

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

4.631.5.46 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (unsigned long long __n) `[inline]`, `[inherited]`

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file ostream.

4.631.5.47 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (double __f) `[inline]`, `[inherited]`

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

4.631.5.48 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (float __f) `[inline]`, `[inherited]`

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

4.631.5.49 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (long double __f) `[inline]`, `[inherited]`

Floating point arithmetic inserters.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

4.631.5.50 template<typename _CharT, typename _Traits> __ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (const void * *__p*) [inline], [inherited]

Pointer arithmetic inserters.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

4.631.5.51 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::operator<< (__streambuf_type * *__sb*) [inherited]

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.52 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__istream_type &(*)(__istream_type &) __*pf*) `[inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 120 of file istream.

4.631.5.53 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (__ios_type &(*)(__ios_type &) __*pf*) `[inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 124 of file istream.

4.631.5.54 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (ios_base &(*)(ios_base &) __*pf*) `[inline], [inherited]`

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

Definition at line 131 of file istream.

4.631.5.55 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (bool & __*n*) `[inline], [inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file istream.

4.631.5.56 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (short & __*n*) `[inherited]`

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 114 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.57 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned short & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

4.631.5.58 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (int & __n) [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 159 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.59 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned int & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file istream.

4.631.5.60 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file istream.

4.631.5.61 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file istream.

4.631.5.62 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file istream.

4.631.5.63 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (unsigned long long & __n) [inline], [inherited]

Integer arithmetic extractors.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file istream.

4.631.5.64 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (float & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file istream.

4.631.5.65 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file istream.

4.631.5.66 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (long double & __f) [inline], [inherited]

Floating point arithmetic extractors.

Parameters

__f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file istream.

4.631.5.67 template<typename _CharT, typename _Traits> __istream_type& std::basic_istream< _CharT, _Traits >::operator>> (void *& __p) [inline], [inherited]

Basic arithmetic extractors.

Parameters

__p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file istream.

4.631.5.68 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::operator>> (__streambuf_type * __sb) [inherited]

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If __`sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the __`sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 204 of file istream.tcc.

References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.69 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::int_type basic_istream< _CharT, _Traits >::peek (void) [inherited]

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 620 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.70 streamsize std::ios_base::precision ( ) const [inline], [inherited]

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.631.5.71 streamsize std::ios_base::precision (streamsize __prec) [inline], [inherited]

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.631.5.72 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::put (char_type __c) [inherited]

Simple insertion.

Parameters

__c	The character to insert.

Returns

∗this

Tries to insert `__c`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

Referenced by std::endl(), and std::ends().

4.631.5.73 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::putback (char_type __c) [inherited]

Unextracting a single character.

Parameters

__c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 711 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

Referenced by std::operator>>().

4.631.5.74 void∗& std::ios_base::pword (int __ix) [inline], [inherited]

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.631.5.75 template<typename _CharT, typename _Traits> basic_streambuf< _CharT, _Traits > ∗ basic_ios< _CharT, _Traits >::rdbuf (basic_streambuf< _CharT, _Traits > ∗ __sb) [inherited]

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

Definition at line 53 of file basic_ios.tcc.

4.631.5.76 template<typename _CharT , typename _Traits , typename _Alloc > __stringbuf_type∗ std::basic_stringstream< _CharT, _Traits, _Alloc >::rdbuf ( ) const [inline]

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

Definition at line 575 of file sstream.

4.631.5.77 template<typename _CharT, typename _Traits> iostate std::basic_ios< _CharT, _Traits >::rdstate ( ) const [inline], [inherited]

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 131 of file basic_ios.h.

Referenced by std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::eof(), std::basic_ios< char, _Traits >::fail(), std::basic_ios< char, _Traits >::good(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char, _Traits >::setstate(), and std::basic_istream< _CharT, _Traits >::unget().

4.631.5.78 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits > & basic_istream< _CharT, _Traits >::read (char_type * __s, streamsize __n) [inherited]

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 650 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.79 template<typename _CharT , typename _Traits > streamsize basic_istream< _CharT, _Traits >::readsome (char_type * __s, streamsize __n) [inherited]

Extraction until the buffer is exhausted, but no more.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 679 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.80 void std::ios_base::register_callback (**event_callback** *__fn,* int *__index*) `[inherited]`

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.631.5.81 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (pos_type *__pos*) `[inherited]`

Changing the current read position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 845 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.82 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::seekg (off_type *__off*, **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 884 of file istream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.83 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (pos_type *__pos*) `[inherited]`

Changing the current write position.

Parameters

__pos	A file position object.

Returns

∗this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.84 template<typename _CharT , typename _Traits > **basic_ostream**< _CharT, _Traits > & **basic_ostream**< _CharT, _Traits >::seekp (off_type *__off*, **ios_base::seekdir** *__dir*) `[inherited]`

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

$*$this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

Definition at line 290 of file ostream.tcc.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.85 fmtflags std::ios_base::setf (fmtflags *__fmtfl*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.631.5.86 fmtflags std::ios_base::setf (fmtflags *__fmtfl,* fmtflags *__mask*) `[inline]`, `[inherited]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.631.5.87 template<typename _CharT, typename _Traits> void std::basic_ios< _CharT, _Traits >::setstate (iostate *__state*) `[inline]`, `[inherited]`

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

Definition at line 151 of file basic_ios.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::tr2::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.631.5.88 template<typename _CharT , typename _Traits , typename _Alloc > __string_type std::basic_stringstream< _CharT, _Traits, _Alloc >::str ( ) const [inline]

Copying out the string buffer.

Returns

`rdbuf()->str()`

Definition at line 583 of file sstream.

4.631.5.89 template<typename _CharT , typename _Traits , typename _Alloc > void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (const __string_type & __s) [inline]

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

Definition at line 593 of file sstream.

4.631.5.90 template<typename _CharT , typename _Traits > int basic_istream< _CharT, _Traits >::sync (void) [inherited]

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.

Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 781 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

4.631.5.91 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`, `[inherited]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.631.5.92 template<typename _CharT , typename _Traits > basic_istream< _CharT, _Traits >::pos_type basic_istream< _CharT, _Traits >::tellg (void) `[inherited]`

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 817 of file istream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.631.5.93 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits >::pos_type basic_ostream< _CharT, _Traits >::tellp ( ) `[inherited]`

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.

Definition at line 237 of file ostream.tcc.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf().

4.631.5.94 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie ( ) const `[inline]`, `[inherited]`

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 289 of file basic_ios.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.631.5.95 template<typename _CharT, typename _Traits> **basic_ostream**<_CharT, _Traits>* **std::basic_ios**< _CharT, _Traits >::tie (**basic_ostream**< _CharT, _Traits > * *__tiestr*) `[inline]`, `[inherited]`

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 301 of file basic_ios.h.

4.631.5.96 template<typename _CharT , typename _Traits > **basic_istream**< _CharT, _Traits > & **basic_istream**< _CharT, _Traits >::unget (void) `[inherited]`

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 746 of file istream.tcc.

References std::basic_istream< _CharT, _Traits >::_M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().

4.631.5.97 void std::ios_base::unsetf (**fmtflags** *__mask*) [inline], [inherited]

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.631.5.98 template<typename _CharT, typename _Traits> char_type **std::basic_ios**< _CharT, _Traits >::widen (char *__c*) const [inline], [inherited]

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at `http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html`

Definition at line 443 of file basic_ios.h.

Referenced by std::endl(), std::basic_ios< char, _Traits >::fill(), std::basic_istream< char >::get(), std::basic_istream< char >::getline(), std::getline(), std::tr2::operator>>(), and std::operator>>().

4.631.5.99 **streamsize** std::ios_base::width ( ) const [inline], [inherited]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.631.5.100 **streamsize** std::ios_base::width (**streamsize** *__wide*) [inline], [inherited]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.631.5.101 template<typename _CharT , typename _Traits > basic_ostream< _CharT, _Traits > & basic_ostream< _CharT, _Traits >::write (const char_type * __s, streamsize __n) [inherited]

Character string insertion.

Parameters

__s	The array to insert.
__n	Maximum number of characters to insert.

Returns

*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

References std::basic_ostream< _CharT, _Traits >::_M_write(), and std::ios_base::badbit.

4.631.5.102 static int std::ios_base::xalloc ( ) throw () [static], [inherited]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.631.6 Member Data Documentation

4.631.6.1 template<typename _CharT, typename _Traits> streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]

The number of characters extracted in the previous unformatted function; see gcount().

Definition at line 82 of file istream.

Referenced by std::basic_istream< char >::gcount(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::unget(), and std::basic_istream< char >::~basic_istream().

4.631.6.2 const fmtflags std::ios_base::adjustfield [static], [inherited]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.631.6.3 const openmode std::ios_base::app [static], [inherited]

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.631.6.4 const openmode std::ios_base::ate [static], [inherited]

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.631.6.5 const iostate std::ios_base::badbit [static], [inherited]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.631.6.6 const fmtflags std::ios_base::basefield [static], [inherited]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream<

_CharT, _Traits >::operator<<().

4.631.6.7 const seekdir std::ios_base::beg [static], [inherited]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.631.6.8 const openmode std::ios_base::binary [static], [inherited]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.631.6.9 const fmtflags std::ios_base::boolalpha [static], [inherited]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.631.6.10 const seekdir std::ios_base::cur [static], [inherited]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.631.6.11 const fmtflags std::ios_base::dec [static], [inherited]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.631.6.12 const seekdir std::ios_base::end [static], [inherited]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.631.6.13 const iostate std::ios_base::eofbit [static], [inherited]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time-

_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.631.6.14 const **iostate** std::ios_base::failbit `[static]`, `[inherited]`

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.631.6.15 const **fmtflags** std::ios_base::fixed `[static]`, `[inherited]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.631.6.16 const **fmtflags** std::ios_base::floatfield `[static]`, `[inherited]`

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.631.6.17 const **iostate** std::ios_base::goodbit `[static]`, `[inherited]`

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.631.6.18 const fmtflags std::ios_base::hex `[static]`, `[inherited]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.631.6.19 const openmode std::ios_base::in `[static]`, `[inherited]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.631.6.20 const fmtflags std::ios_base::internal `[static]`, `[inherited]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.631.6.21 const fmtflags std::ios_base::left `[static]`, `[inherited]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.631.6.22 const fmtflags std::ios_base::oct `[static]`, `[inherited]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.631.6.23 const openmode std::ios_base::out `[static]`, `[inherited]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.631.6.24 const fmtflags std::ios_base::right `[static]`, `[inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.631.6.25 const **fmtflags** std::ios_base::scientific `[static], [inherited]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.631.6.26 const **fmtflags** std::ios_base::showbase `[static], [inherited]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.631.6.27 const **fmtflags** std::ios_base::showpoint `[static], [inherited]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.631.6.28 const **fmtflags** std::ios_base::showpos `[static], [inherited]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.631.6.29 const **fmtflags** std::ios_base::skipws `[static], [inherited]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.631.6.30 const **openmode** std::ios_base::trunc `[static], [inherited]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.631.6.31 const **fmtflags** std::ios_base::unitbuf `[static], [inherited]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.631.6.32 const **fmtflags** std::ios_base::uppercase `[static], [inherited]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- sstream

4.632 std::bernoulli_distribution Class Reference

Classes

- struct param_type

Public Types

- typedef bool result_type

Public Member Functions

- bernoulli_distribution (double __p=0.5)
- **bernoulli_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const bernoulli_distribution &__d1, const bernoulli_distribution &__d2)

4.632.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood p that true will come up and $(1-p)$ that false will appear.

Definition at line 3572 of file random.h.

4.632.2 Member Typedef Documentation

4.632.2.1 typedef bool **std::bernoulli_distribution::result_type**

The type of the range of the distribution.

Definition at line 3576 of file random.h.

4.632.3 Constructor & Destructor Documentation

4.632.3.1 std::bernoulli_distribution::bernoulli_distribution (double *__p* = `0.5`) `[inline]`, `[explicit]`

Constructs a Bernoulli distribution with likelihood `p`.

Parameters

__p	[IN] The likelihood of a true result being returned. Must be in the interval $[0,1]$.

Definition at line 3609 of file random.h.

4.632.4 Member Function Documentation

4.632.4.1 **result_type** std::bernoulli_distribution::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3659 of file random.h.

References std::numeric_limits< _Tp >::max().

4.632.4.2 **result_type** std::bernoulli_distribution::min ( ) const `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3652 of file random.h.

References std::numeric_limits< _Tp >::min().

4.632.4.3 template<typename _UniformRandomNumberGenerator > **result_type** std::bernoulli_distribution::operator() (_UniformRandomNumberGenerator & *__urng*) `[inline]`

Generating functions.

Definition at line 3667 of file random.h.

References operator()().

Referenced by operator()().

4.632.4.4 double std::bernoulli_distribution::p ( ) const `[inline]`

Returns the `p` parameter of the distribution.

Definition at line 3630 of file random.h.

4.632.4.5 **param_type** std::bernoulli_distribution::param ( ) const `[inline]`

Returns the parameter set of the distribution.

Definition at line 3637 of file random.h.

Referenced by std::operator>>().

4.632.4.6 void std::bernoulli_distribution::param (const param_type & __param) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 3645 of file random.h.

4.632.4.7 void std::bernoulli_distribution::reset ( ) [inline]

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3624 of file random.h.

4.632.5 Friends And Related Function Documentation

4.632.5.1 bool operator== (const bernoulli_distribution & *__d1*, const bernoulli_distribution & *__d2*) [friend]

Return true if two Bernoulli distributions have the same parameters.

Definition at line 3709 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.633 std::bernoulli_distribution::param_type Struct Reference

Public Types

- typedef bernoulli_distribution **distribution_type**

Public Member Functions

- **param_type** (double __p=0.5)
- double **p** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.633.1 Detailed Description

Parameter type.

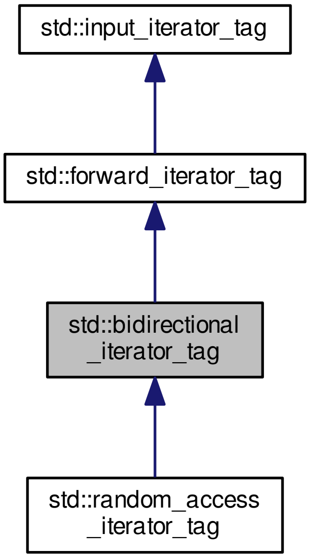
Definition at line 3578 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.634 std::bidirectional_iterator_tag Struct Reference

Inheritance diagram for std::bidirectional_iterator_tag:



4.634.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.

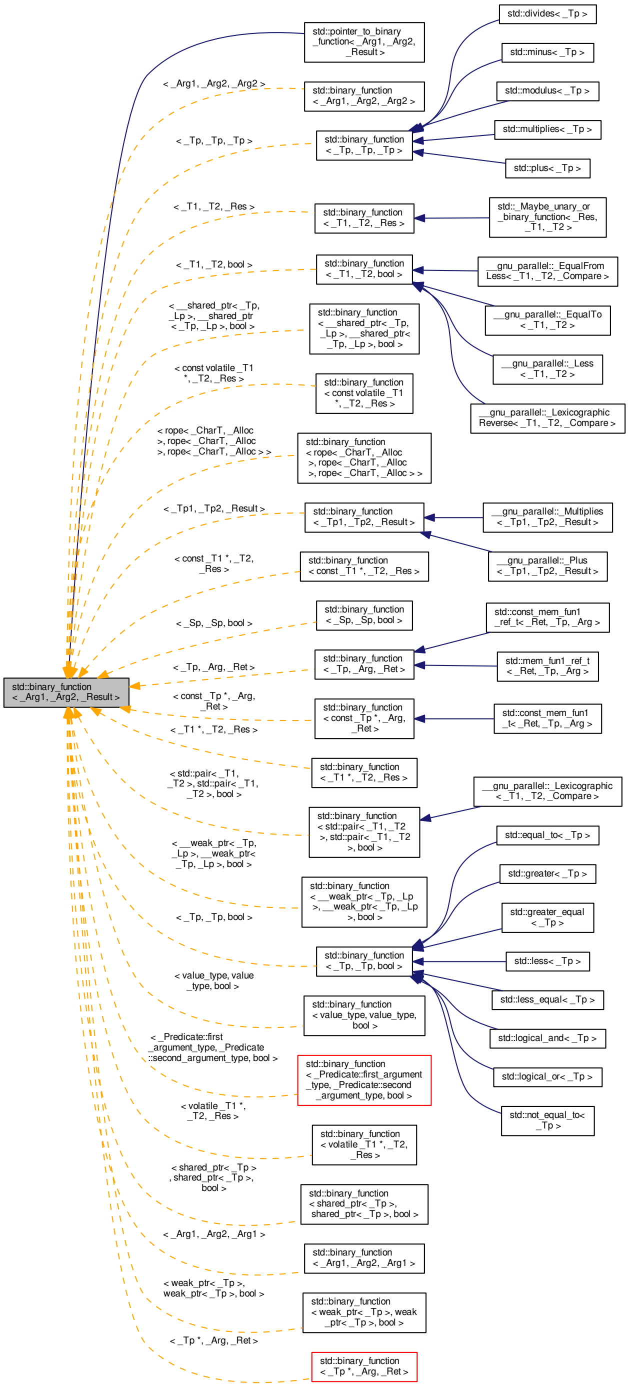
Definition at line 99 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.635 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference

Inheritance diagram for std::binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

4.635.1 Detailed Description

template<typename _Arg1, typename _Arg2, typename _Result>struct std::binary_function< _Arg1, _Arg2, _Result >

This is one of the functor base classes.

Definition at line 114 of file stl_function.h.

4.635.2 Member Typedef Documentation

4.635.2.1 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.635.2.2 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.635.2.3 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type

`second_argument_type` is the type of the second argument

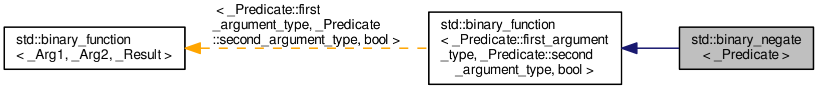
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.636 std::binary_negate< _Predicate > Class Template Reference

Inheritance diagram for std::binary_negate< _Predicate >:



Public Types

- typedef
  _Predicate::first_argument_type first_argument_type
- typedef bool result_type
- typedef
  _Predicate::second_argument_type second_argument_type

Public Member Functions

- **binary_negate** (const _Predicate &__x)
- bool **operator()** (const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y) const

Protected Attributes

- _Predicate **_M_pred**

4.636.1 Detailed Description

template<typename _Predicate>class std::binary_negate< _Predicate >

One of the negation functors.

Definition at line 374 of file stl_function.h.

4.636.2 Member Typedef Documentation

4.636.2.1 typedef _Predicate::first_argument_type **std::binary_function**< _Predicate::first_argument_type , _Predicate::second_argument_type , bool >**::first_argument_type** [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.636.2.2 typedef bool **std::binary_function**< _Predicate::first_argument_type , _Predicate::second_argument_type , bool >**::result_type** [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.636.2.3 typedef _Predicate::second_argument_type **std::binary_function**< _Predicate::first_argument_type , _Predicate::second_argument_type , bool >**::second_argument_type** [inherited]

`second_argument_type` is the type of the second argument

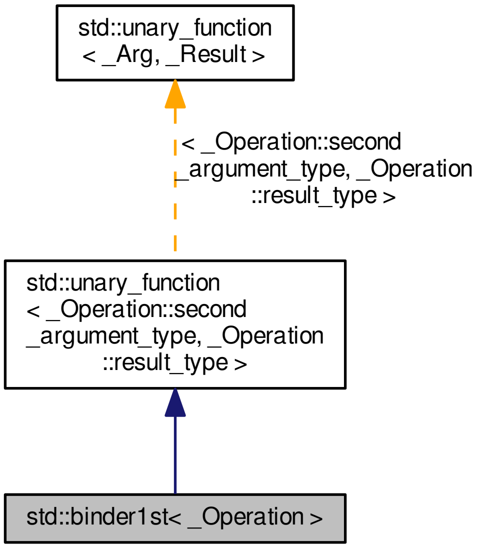
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.637 std::binder1st< _Operation > Class Template Reference

Inheritance diagram for std::binder1st< _Operation >:



Public Types

- typedef _Operation::second_argument_type argument_type
- typedef _Operation::result_type result_type

Public Member Functions

- **binder1st** (const _Operation &__x, const typename _Operation::first_argument_type &__y)
- _Operation::result_type **operator()** (const typename _Operation::second_argument_type &__x) const
- _Operation::result_type **operator()** (typename _Operation::second_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::first_argument_type **value**

4.637.1 Detailed Description

template<typename _Operation>class std::binder1st< _Operation >

One of the binder functors.

Definition at line 104 of file binders.h.

4.637.2 Member Typedef Documentation

4.637.2.1 typedef _Operation::second_argument_type **std::unary_function**< _Operation::second_argument_type , _Operation::result_type >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.637.2.2 typedef _Operation::result_type **std::unary_function**< _Operation::second_argument_type , _Operation::result_type >**::result_type** `[inherited]`

`result_type` is the return type

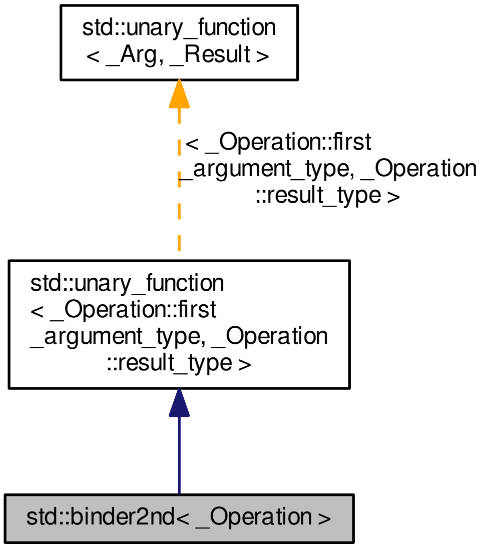
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- binders.h

4.638 std::binder2nd< _Operation > Class Template Reference

Inheritance diagram for std::binder2nd< _Operation >:



Public Types

- typedef
  _Operation::first_argument_type argument_type
- typedef _Operation::result_type result_type

Public Member Functions

- **binder2nd** (const _Operation &__x, const typename _Operation::second_argument_type &__y)
- _Operation::result_type **operator()** (const typename _Operation::first_argument_type &__x) const
- _Operation::result_type **operator()** (typename _Operation::first_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::second_argument_type **value**

4.638.1 Detailed Description

template<typename _Operation>class std::binder2nd< _Operation >

One of the binder functors.

Definition at line 139 of file binders.h.

4.638.2 Member Typedef Documentation

4.638.2.1 typedef _Operation::first_argument_type **std::unary_function**< _Operation::first_argument_type , _Operation::result_type >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.638.2.2 typedef _Operation::result_type **std::unary_function**< _Operation::first_argument_type , _Operation::result_type >**::result_type** `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- binders.h

4.639 std::binomial_distribution< _IntType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **binomial_distribution** (_IntType __t=_IntType(1), double __p=0.5)
- **binomial_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _IntType t () const

Friends

- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::binomial_distribution< _IntType1 > &__x)
- bool operator== (const binomial_distribution &__d1, const binomial_distribution &__d2)
- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::binomial_distribution< _IntType1 > &__x)

4.639.1 Detailed Description

template<typename _IntType = int>class std::binomial_distribution< _IntType >

A discrete binomial random number distribution.

The formula for the binomial probability density function is $p(i|t,p) = \binom{t}{i} p^i (1 - p)^{t - i}$ where t and p are the parameters of the distribution.

Definition at line 3777 of file random.h.

4.639.2 Member Typedef Documentation

4.639.2.1 template<typename _IntType = int> typedef _IntType **std::binomial_distribution**< _IntType >::**result_type**

The type of the range of the distribution.

Definition at line 3780 of file random.h.

4.639.3 Member Function Documentation

4.639.3.1 template<typename _IntType = int> **result_type std::binomial_distribution**< _IntType >::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3887 of file random.h.

4.639.3.2 template<typename _IntType = int> **result_type std::binomial_distribution**< _IntType >::min ( ) const `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3880 of file random.h.

4.639.3.3 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > **result_type std::binomial_distribution**< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*) `[inline]`

Generating functions.

Definition at line 3895 of file random.h.

References std::binomial_distribution< _IntType >::operator()().

Referenced by std::binomial_distribution< _IntType >::operator()().

4.639.3.4 template<typename _IntType > template<typename _UniformRandomNumberGenerator > **binomial_distribution**< _IntType >::**result_type std::binomial_distribution**< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*, const **param_type** & *__param*)

A rejection algorithm when $t * p >= 8$ and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if _GLIBCXX_USE_C99_MATH_TR1 is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1685 of file bits/random.tcc.

References std::abs(), std::numeric_limits< _Tp >::epsilon(), std::log(), and std::numeric_limits< _Tp >::max().

4.639.3.5 template<typename _IntType = int> double **std::binomial_distribution**< _IntType >::p ( ) const `[inline]`

Returns the distribution `p` parameter.

Definition at line 3858 of file random.h.

4.639.3.6 template<typename _IntType = int> **param_type std::binomial_distribution**< _IntType >::param ( ) const `[inline]`

Returns the parameter set of the distribution.

Definition at line 3865 of file random.h.

4.639.3.7 template<typename _IntType = int> void **std::binomial_distribution**< _IntType >::param (const **param_type** & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 3873 of file random.h.

4.639.3.8 template<typename _IntType = int> void **std::binomial_distribution**< _IntType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 3844 of file random.h.

References std::normal_distribution< _RealType >::reset().

4.639.3.9 template<typename _IntType = int> _IntType **std::binomial_distribution**< _IntType >::t ( ) const [inline]

Returns the distribution `t` parameter.

Definition at line 3851 of file random.h.

4.639.4 Friends And Related Function Documentation

4.639.4.1 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > **std::basic_ostream**<_CharT, _Traits>& operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os,* const **std::binomial_distribution**< _IntType1 > & *__x*) [friend]

Inserts a binomial_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A binomial_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.639.4.2 template<typename _IntType = int> bool operator== (const **binomial_distribution**< _IntType > & *__d1,* const **binomial_distribution**< _IntType > & *__d2*) [friend]

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3931 of file random.h.

4.639.4.3 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > **std::basic_istream**<_CharT, _Traits>& operator>> (**std::basic_istream**< _CharT, _Traits > & *__is,* **std::binomial_distribution**< _IntType1 > & *__x*) [friend]

Extracts a binomial_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A binomial_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.640 std::binomial_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef binomial_distribution
  < _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __t=_IntType(1), double __p=0.5)
- double **p** () const
- _IntType **t** () const

Friends

- class **binomial_distribution**< **_IntType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.640.1 Detailed Description

template<typename _IntType = int>struct std::binomial_distribution< _IntType >::param_type

Parameter type.

Definition at line 3786 of file random.h.

The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

4.641 std::cauchy_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **cauchy_distribution** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **cauchy_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **a** () const
- _RealType **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const cauchy_distribution &__d1, const cauchy_distribution &__d2)

4.641.1 Detailed Description

template<typename _RealType = double>class std::cauchy_distribution< _RealType >

A cauchy_distribution random number distribution.

The formula for the normal probability mass function is $p(x|a,b) = (\pi b(1+(\frac{x-a}{b})^2))^{-1}$

Definition at line 2929 of file random.h.

4.641.2 Member Typedef Documentation

4.641.2.1 template<typename _RealType = double> typedef _RealType std::cauchy_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 2932 of file random.h.

4.641.3 Member Function Documentation

4.641.3.1 template<typename _RealType = double> **result_type std::cauchy_distribution**< _RealType >::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 3020 of file random.h.

References std::numeric_limits< _Tp >::max().

4.641.3.2 template<typename _RealType = double> **result_type std::cauchy_distribution**< _RealType >::min ( ) const `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 3013 of file random.h.

References std::numeric_limits< _Tp >::min().

4.641.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > **result_type std::cauchy_distribution**< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) `[inline]`

Generating functions.

Definition at line 3028 of file random.h.

References std::cauchy_distribution< _RealType >::operator()().

Referenced by std::cauchy_distribution< _RealType >::operator()().

4.641.3.4 template<typename _RealType = double> **param_type std::cauchy_distribution**< _RealType >::param ( ) const `[inline]`

Returns the parameter set of the distribution.

Definition at line 2998 of file random.h.

Referenced by std::operator>>().

4.641.3.5 template<typename _RealType = double> void **std::cauchy_distribution**< _RealType >::param (const **param_type** & *__param*) `[inline]`

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 3006 of file random.h.

4.641.3.6 template<typename _RealType = double> void **std::cauchy_distribution**< _RealType >::reset ( ) `[inline]`

Resets the distribution state.

Definition at line 2980 of file random.h.

4.641.4 Friends And Related Function Documentation

4.641.4.1 template<typename _RealType = double> bool operator== (const cauchy_distribution< _RealType > & *__d1*, const cauchy_distribution< _RealType > & *__d2*) [friend]

Return true if two Cauchy distributions have the same parameters.

Definition at line 3063 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.642 std::cauchy_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef cauchy_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.642.1 Detailed Description

template<typename _RealType = double>struct std::cauchy_distribution< _RealType >::param_type

Parameter type.

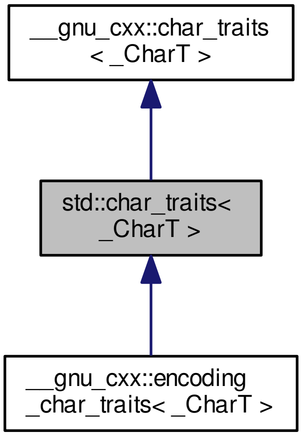
Definition at line 2938 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.643 std::char_traits< _CharT > Struct Template Reference

Inheritance diagram for std::char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- typedef _Char_types< _CharT > ::int_type **int_type**
- typedef _Char_types< _CharT > ::off_type **off_type**
- typedef _Char_types< _CharT > ::pos_type **pos_type**
- typedef _Char_types< _CharT > ::state_type **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type ∗ **assign** (char_type ∗__s, std::size_t __n, char_type __a)
- static int **compare** (const char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static char_type ∗ **copy** (char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type ∗ **find** (const char_type ∗__s, std::size_t __n, const char_type &__a)
- static std::size_t **length** (const char_type ∗__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type ∗ **move** (char_type ∗__s1, const char_type ∗__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)

- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

4.643.1 Detailed Description

template<class _CharT>struct std::char_traits< _CharT >

Basis for explicit traits specializations.

Note

> For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around __gnu_cxx::char_traits, it is possible to achieve a more appropriate definition by specializing __gnu_cxx::char_traits.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt05ch13s03.html` for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod_char_traits.h.

Definition at line 227 of file char_traits.h.

The documentation for this struct was generated from the following file:

- char_traits.h

4.644 std::char_traits< __gnu_cxx::character< V, I, S > > Struct Template Reference

Public Types

- typedef __gnu_cxx::character
  < V, I, S > **char_type**
- typedef char_type::int_type **int_type**
- typedef streamoff **off_type**
- typedef fpos< state_type > **pos_type**
- typedef char_type::state_type **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type ∗ **assign** (char_type ∗__s, size_t __n, char_type __a)
- static int **compare** (const char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static char_type ∗ **copy** (char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static int_type **eof** ()
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static const char_type ∗ **find** (const char_type ∗__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type ∗__s)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static char_type ∗ **move** (char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static int_type **not_eof** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__i)
- static int_type **to_int_type** (const char_type &__c)

4.644.1 Detailed Description

template<typename V, typename I, typename S>struct std::char_traits< __gnu_cxx::character< V, I, S > >

char_traits<__gnu_cxx::character> specialization.

Definition at line 95 of file pod_char_traits.h.

The documentation for this struct was generated from the following file:

- pod_char_traits.h

4.645 std::char_traits< char > Struct Template Reference

Public Types

- typedef char **char_type**
- typedef int **int_type**
- typedef streamoff **off_type**
- typedef streampos **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static char_type ∗ **assign** (char_type ∗__s, size_t __n, char_type __a)
- static int **compare** (const char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static char_type ∗ **copy** (char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static const char_type ∗ **find** (const char_type ∗__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type ∗__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static char_type ∗ **move** (char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

4.645.1 Detailed Description

template<>struct std::char_traits< char >

21.1.3.1 char_traits specializations

Definition at line 233 of file char_traits.h.

The documentation for this struct was generated from the following file:

- char_traits.h

4.646 std::char_traits< wchar_t > Struct Template Reference

Public Types

- typedef wchar_t **char_type**
- typedef wint_t **int_type**
- typedef streamoff **off_type**
- typedef wstreampos **pos_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static char_type ∗ **assign** (char_type ∗__s, size_t __n, char_type __a)
- static int **compare** (const char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static char_type ∗ **copy** (char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static const char_type ∗ **find** (const char_type ∗__s, size_t __n, const char_type &__a)
- static size_t **length** (const char_type ∗__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static char_type ∗ **move** (char_type ∗__s1, const char_type ∗__s2, size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

4.646.1 Detailed Description

template<>struct std::char_traits< wchar_t >

21.1.3.2 char_traits specializations

Definition at line 304 of file char_traits.h.

The documentation for this struct was generated from the following file:

- char_traits.h

4.647 std::chi_squared_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **chi_squared_distribution** (_RealType __n=_RealType(1))
- **chi_squared_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::chi_squared_distribution< _RealType1 > &__x)
- bool operator== (const chi_squared_distribution &__d1, const chi_squared_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::chi_squared_distribution< _RealType1 > &__x)

4.647.1 Detailed Description

template<typename _RealType = double>class std::chi_squared_distribution< _RealType >

A chi_squared_distribution random number distribution.

The formula for the normal probability mass function is $p(x|n) = \frac{x^{(n/2)-1}e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

Definition at line 2719 of file random.h.

4.647.2 Member Typedef Documentation

4.647.2.1 template<typename _RealType = double> typedef _RealType **std::chi_squared_distribution**< _RealType >::**result_type**

The type of the range of the distribution.

Definition at line 2722 of file random.h.

4.647.3 Member Function Documentation

4.647.3.1 template<typename _RealType = double> **result_type std::chi_squared_distribution**< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 2799 of file random.h.

References std::numeric_limits< _Tp >::max().

4.647.3.2 template<typename _RealType = double> **result_type std::chi_squared_distribution**< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 2792 of file random.h.

4.647.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > **result_type std::chi_squared_distribution**< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 2807 of file random.h.

4.647.3.4 template<typename _RealType = double> **param_type std::chi_squared_distribution**< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 2777 of file random.h.

4.647.3.5 template<typename _RealType = double> void **std::chi_squared_distribution**< _RealType >::param (const **param_type** & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 2785 of file random.h.

4.647.3.6 template<typename _RealType = double> void **std::chi_squared_distribution**< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 2763 of file random.h.

References std::gamma_distribution< _RealType >::reset().

4.647.4 Friends And Related Function Documentation

4.647.4.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & *__os*, const std::chi_squared_distribution< _RealType1 > & *__x*) [friend]

Inserts a chi_squared_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A chi_squared_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.647.4.2 template<typename _RealType = double> bool operator== (const chi_squared_distribution< _RealType > & *__d1*, const chi_squared_distribution< _RealType > & *__d2*) [friend]

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2858 of file random.h.

4.647.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & *__is*, std::chi_squared_distribution< _RealType1 > & *__x*) [friend]

Extracts a chi_squared_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A chi_squared_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.648 std::chi_squared_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef chi_squared_distribution < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n=_RealType(1))
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.648.1 Detailed Description

template<typename _RealType = double>struct std::chi_squared_distribution< _RealType >::param_type

Parameter type.

Definition at line 2728 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.649 std::chrono::duration< _Rep, _Period > Struct Template Reference

Public Types

- typedef _Period **period**
- typedef _Rep **rep**

Public Member Functions

- **duration** (const duration &)=default
- template<typename _Rep2 , typename = typename enable_if<is_convertible<_Rep2, rep>::value && (treat_as_floating_point<rep>::value || !treat_as_floating_point<_Rep2>::value)>::type>
  constexpr **duration** (const _Rep2 &__rep)
- template<typename _Rep2 , typename _Period2 , typename = typename enable_if<treat_as_floating_point<rep>::value || (ratio_divide<_Period2, period>::den == 1 && !treat_as_floating_point<_Rep2>::value)>::type>
  constexpr **duration** (const duration< _Rep2, _Period2 > &__d)
- constexpr rep **count** () const
- template<typename _Rep2 = rep>
  enable_if
  <!treat_as_floating_point
  < _Rep2 >::value, duration & >
  ::type **operator%=** (const rep &__rhs)
- template<typename _Rep2 = rep>
  enable_if
  <!treat_as_floating_point
  < _Rep2 >::value, duration & >
  ::type **operator%=** (const duration &__d)
- duration & **operator*=** (const rep &__rhs)
- constexpr duration **operator+** () const
- duration & **operator++** ()

- duration **operator++** (int)
- duration & **operator+=** (const duration &__d)
- constexpr duration **operator-** () const
- duration & **operator--** ()
- duration **operator--** (int)
- duration & **operator-=** (const duration &__d)
- duration & **operator/=** (const rep &__rhs)
- duration & **operator=** (const duration &)=default

Static Public Member Functions

- static constexpr duration **max** ()
- static constexpr duration **min** ()
- static constexpr duration **zero** ()

4.649.1 Detailed Description

template<typename _Rep, typename _Period>struct std::chrono::duration< _Rep, _Period >

duration

Definition at line 240 of file chrono.

The documentation for this struct was generated from the following file:

- chrono

4.650 std::chrono::duration_values< _Rep > Struct Template Reference

Static Public Member Functions

- static constexpr _Rep **max** ()
- static constexpr _Rep **min** ()
- static constexpr _Rep **zero** ()

4.650.1 Detailed Description

template<typename _Rep>struct std::chrono::duration_values< _Rep >

duration_values

Definition at line 213 of file chrono.

The documentation for this struct was generated from the following file:

- chrono

4.651 std::chrono::system_clock Struct Reference

Public Types

- typedef chrono::seconds **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef chrono::time_point
  < system_clock, duration > **time_point**

Static Public Member Functions

- static time_point **from_time_t** (std::time_t __t) noexcept
- static time_point **now** () noexcept
- static std::time_t **to_time_t** (const time_point &__t) noexcept

Static Public Attributes

- static constexpr bool **is_steady**

4.651.1 Detailed Description

system_clock

Definition at line 690 of file chrono.

The documentation for this struct was generated from the following file:

- chrono

4.652 std::chrono::time_point< _Clock, _Dur > Struct Template Reference

Public Types

- typedef _Clock **clock**
- typedef _Dur **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**

Public Member Functions

- constexpr **time_point** (const duration &__dur)
- template<typename _Dur2 >
  constexpr **time_point** (const time_point< clock, _Dur2 > &__t)
- time_point & **operator+=** (const duration &__dur)
- time_point & **operator-=** (const duration &__dur)
- constexpr duration **time_since_epoch** () const

Static Public Member Functions

- static constexpr time_point **max** ()
- static constexpr time_point **min** ()

4.652.1 Detailed Description

template<typename _Clock, typename _Dur>struct std::chrono::time_point< _Clock, _Dur >

time_point

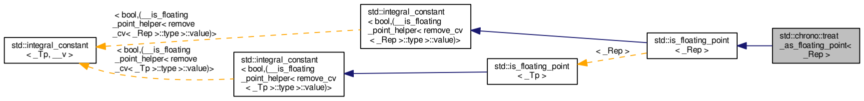
Definition at line 545 of file chrono.

The documentation for this struct was generated from the following file:

- chrono

4.653 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference

Inheritance diagram for std::chrono::treat_as_floating_point< _Rep >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.653.1 Detailed Description

template<typename _Rep>struct std::chrono::treat_as_floating_point< _Rep >

treat_as_floating_point

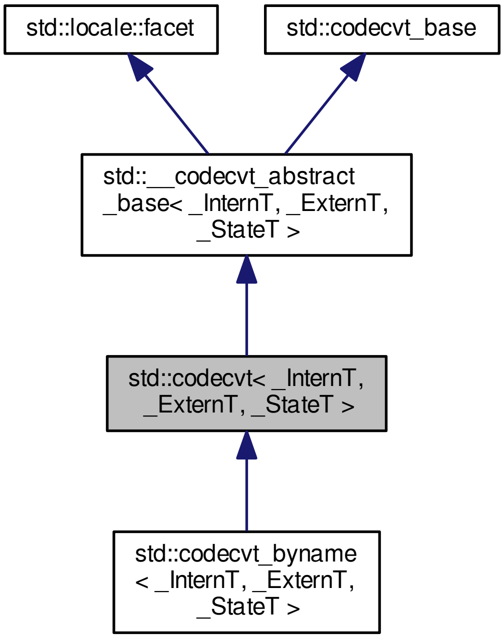
Definition at line 207 of file chrono.

The documentation for this struct was generated from the following file:

- chrono

4.654 std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference

Inheritance diagram for std::codecvt< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- int **length** (state_type &__state, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- result unshift (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.654.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>class std::codecvt< _InternT, _ExternT, _StateT >

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 276 of file codecvt.h.

4.654.2 Member Function Documentation

4.654.2.1 template<typename _InternT , typename _ExternT , typename _StateT > virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & *__state,* const intern_type * *__from,* const intern_type * *__from_end,* const intern_type *& *__from_next,* extern_type * *__to,* extern_type * *__to_end,* extern_type *& *__to_next*) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See Also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

4.654.2.2 template<typename _InternT, typename _ExternT, typename _StateT> result **std::__codecvt_abstract_base**< _InternT, _ExternT, _StateT >::in (state_type & *__state,* const extern_type ∗ *__from,* const extern_type ∗ *__from_end,* const extern_type ∗& *__from_next,* intern_type ∗ *__to,* intern_type ∗ *__to_end,* intern_type ∗& *__to_next*) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.654.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result **std::__codecvt_abstract_base**< _InternT, _ExternT, _StateT >::out (state_type & *__state,* const intern_type ∗ *__from,* const intern_type ∗ *__from_end,* const intern_type ∗& *__from_next,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.654.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (state_type & *__state,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

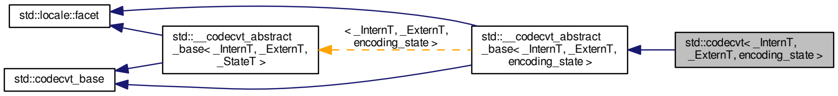
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt.h

4.655 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference

Inheritance diagram for std::codecvt< _InternT, _ExternT, encoding_state >:



Public Types

- typedef state_type::descriptor_type **descriptor_type**
- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef __gnu_cxx::encoding_state **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (state_type &__enc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- int **length** (state_type &__state, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- result unshift (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- virtual int **do_length** (state_type &, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result do_out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.655.1 Detailed Description

template<typename _InternT, typename _ExternT>class std::codecvt< _InternT, _ExternT, encoding_state >

codecvt<InternT, _ExternT, encoding_state> specialization.

Definition at line 230 of file codecvt_specializations.h.

4.655.2 Member Function Documentation

4.655.2.1 template<typename _InternT , typename _ExternT > codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_out (state_type & *__state*, const intern_type ∗ *__from*, const intern_type ∗ *__from_end*, const intern_type ∗& *__from_next*, extern_type ∗ *__to*, extern_type ∗ *__to_end*, extern_type ∗& *__to_next*) const `[protected]`, `[virtual]`

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See Also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

Definition at line 306 of file codecvt_specializations.h.

4.655.2.2 result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (state_type & *__state*, const extern_type ∗ *__from*, const extern_type ∗ *__from_end*, const extern_type ∗& *__from_next*, intern_type ∗ *__to*, intern_type ∗ *__to_end*, intern_type ∗& *__to_next*) const `[inline]`, `[inherited]`

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.655.2.3 result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (state_type & *__state,* const intern_type ∗ *__from,* const intern_type ∗ *__from_end,* const intern_type ∗& *__from_next,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

References std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out().

4.655.2.4 result **std::__codecvt_abstract_base**< _InternT, _ExternT, **encoding_state** >::unshift (**state_type** & *__state*, **extern_type** ∗ *__to*, **extern_type** ∗ *__to_end*, **extern_type** ∗& *__to_next*) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

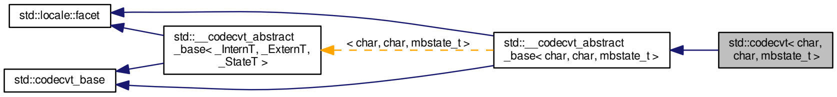
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt_specializations.h

4.656 std::codecvt< char, char, mbstate_t > Class Template Reference

Inheritance diagram for std::codecvt< char, char, mbstate_t >:



Public Types

- typedef char **extern_type**

- typedef char **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- int **length** (state_type &__state, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- result unshift (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- virtual int **do_length** (state_type &, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result do_out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.656.1 Detailed Description

template<>class std::codecvt< char, char, mbstate_t >

class codecvt<char, char, mbstate_t> specialization.

Definition at line 340 of file codecvt.h.

4.656.2 Member Function Documentation

4.656.2.1 virtual result std::codecvt< char, char, mbstate_t >::do_out (state_type & *__state,* const intern_type * *__from,* const intern_type * *__from_end,* const intern_type *& *__from_next,* extern_type * *__to,* extern_type * *__to_end,* extern_type *& *__to_next*) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See Also

out for more information.

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

4.656.2.2 result std::__codecvt_abstract_base< char , char , mbstate_t >::in (state_type & *__state,* const extern_type * *__from,* const extern_type * *__from_end,* const extern_type *& *__from_next,* intern_type * *__to,* intern_type * *__to_end,* intern_type *& *__to_next*) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.656.2.3 result **std::__codecvt_abstract_base**< char , char , mbstate_t >::out (state_type & *__state,* const intern_type ∗ *__from,* const intern_type ∗ *__from_end,* const intern_type ∗& *__from_next,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.656.2.4 result **std::__codecvt_abstract_base**< char , char , mbstate_t >::unshift (state_type & *__state,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

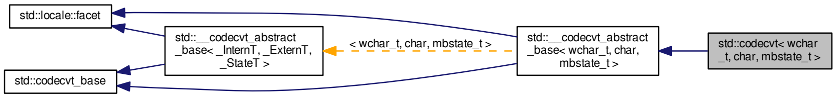
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt.h

4.657 std::codecvt< wchar_t, char, mbstate_t > Class Template Reference

Inheritance diagram for std::codecvt< wchar_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef wchar_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- int **length** (state_type &__state, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- result unshift (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- virtual int **do_length** (state_type &, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result do_out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.657.1 Detailed Description

template<>class std::codecvt< wchar_t, char, mbstate_t >

class codecvt<wchar_t, char, mbstate_t> specialization.

Definition at line 398 of file codecvt.h.

4.657.2 Member Function Documentation

4.657.2.1 virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (state_type & *__state*, const intern_type ∗ *__from*, const intern_type ∗ *__from_end*, const intern_type ∗& *__from_next*, extern_type ∗ *__to*, extern_type ∗ *__to_end*, extern_type ∗& *__to_next*) const [protected], [virtual]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See Also

out for more information.

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

4.657.2.2 result **std::__codecvt_abstract_base**< wchar_t , char , mbstate_t >::in (state_type & *__state,* const extern_type * *__from,* const extern_type * *__from_end,* const extern_type *& *__from_next,* intern_type * *__to,* intern_type * *__to_end,* intern_type *& *__to_next*) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.657.2.3 result **std::__codecvt_abstract_base**< wchar_t , char , mbstate_t >::out (state_type & *__state,* const intern_type * *__from,* const intern_type * *__from_end,* const intern_type *& *__from_next,* extern_type * *__to,* extern_type * *__to_end,* extern_type *& *__to_next*) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.657.2.4 result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::unshift (state_type & *__state,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

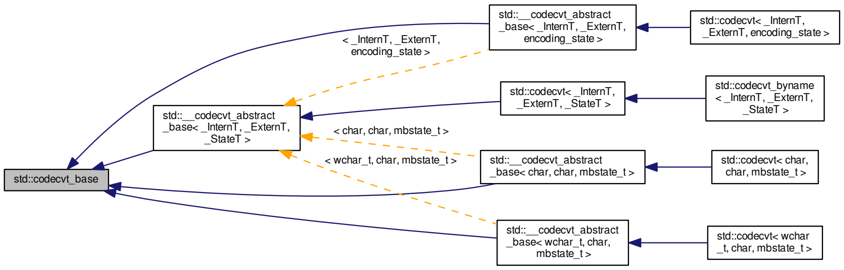
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt.h

4.658 std::codecvt_base Class Reference

Inheritance diagram for std::codecvt_base:



Public Types

- enum **result** { **ok**, **partial**, **error**, **noconv** }

4.658.1 Detailed Description

Empty base class for codecvt facet [22.2.1.5].

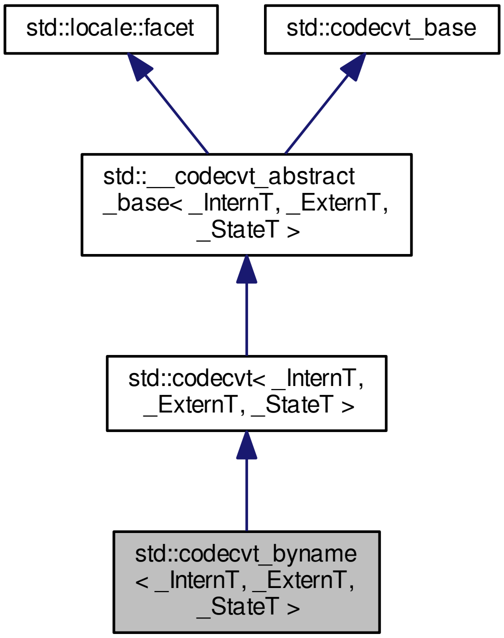
Definition at line 46 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt.h

4.659 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference

Inheritance diagram for std::codecvt_byname< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- **codecvt_byname** (const char ∗__s, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type ∗__from, const extern_type ∗__from_end, const extern_type ∗&__from_next, intern_type ∗__to, intern_type ∗__to_end, intern_type ∗&__to_next) const
- int **length** (state_type &__state, const extern_type ∗__from, const extern_type ∗__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type ∗__from, const intern_type ∗__from_end, const intern_type ∗&__from_next, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const
- result unshift (state_type &__state, extern_type ∗__to, extern_type ∗__to_end, extern_type ∗&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool **do_always_noconv** () const throw ()
- virtual int **do_encoding** () const throw ()
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int **do_max_length** () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

4.659.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>class std::codecvt_byname< _InternT, _ExternT, _StateT >

class codecvt_byname [22.2.1.6].

Definition at line 458 of file codecvt.h.

4.659.2 Member Function Documentation

4.659.2.1 template<typename _InternT , typename _ExternT , typename _StateT > virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (state_type & *__state*, const intern_type * *__from*, const intern_type * *__from_end*, const intern_type *& *__from_next*, extern_type * *__to*, extern_type * *__to_end*, extern_type *& *__to_next*) const `[protected]`, `[virtual]`, `[inherited]`

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See Also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

4.659.2.2 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (state_type & *__state*, const extern_type ∗ *__from*, const extern_type ∗ *__from_end*, const extern_type ∗& *__from_next*, intern_type ∗ *__to*, intern_type ∗ *__to_end*, intern_type ∗& *__to_next*) const [inline], [inherited]

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 196 of file codecvt.h.

4.659.2.3 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (state_type & *__state*, const intern_type ∗ *__from*, const intern_type ∗ *__from_end*, const intern_type ∗& *__from_next*, extern_type ∗ *__to*, extern_type ∗ *__to_end*, extern_type ∗& *__to_next*) const [inline], [inherited]

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

Definition at line 116 of file codecvt.h.

4.659.2.4 template<typename _InternT, typename _ExternT, typename _StateT> result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (state_type & *__state,* extern_type ∗ *__to,* extern_type ∗ *__to_end,* extern_type ∗& *__to_next*) const [inline], [inherited]

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

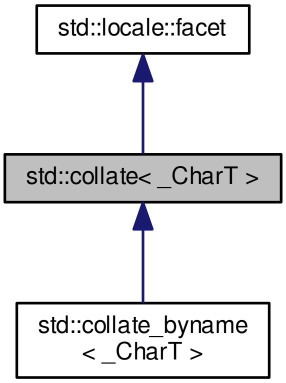
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- codecvt.h

4.660 std::collate< _CharT > Class Template Reference

Inheritance diagram for std::collate< _CharT >:



Public Types

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- collate (size_t __refs=0)
- collate (__c_locale __cloc, size_t __refs=0)
- int **_M_compare** (const _CharT ∗, const _CharT ∗) const throw ()
- template<>
  int **_M_compare** (const char ∗, const char ∗) const throw()
- template<>
  int **_M_compare** (const wchar_t ∗, const wchar_t ∗) const throw()
- size_t **_M_transform** (_CharT ∗, const _CharT ∗, size_t) const throw ()
- template<>
  size_t **_M_transform** (char ∗, const char ∗, size_t) const throw()
- template<>
  size_t **_M_transform** (wchar_t ∗, const wchar_t ∗, size_t) const throw()
- int compare (const _CharT ∗__lo1, const _CharT ∗__hi1, const _CharT ∗__lo2, const _CharT ∗__hi2) const
- long hash (const _CharT ∗__lo, const _CharT ∗__hi) const
- string_type transform (const _CharT ∗__lo, const _CharT ∗__hi) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~collate ()
- virtual int do_compare (const _CharT ∗__lo1, const _CharT ∗__hi1, const _CharT ∗__lo2, const _CharT ∗__hi2) const
- virtual long do_hash (const _CharT ∗__lo, const _CharT ∗__hi) const
- virtual string_type do_transform (const _CharT ∗__lo, const _CharT ∗__hi) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_collate**

4.660.1 Detailed Description

template<typename _CharT>class std::collate< _CharT >

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 584 of file locale_classes.h.

4.660.2 Member Typedef Documentation

4.660.2.1 template<typename _CharT> typedef _CharT std::collate< _CharT >::char_type

Public typedefs.

Definition at line 590 of file locale_classes.h.

4.660.2.2 template<typename _CharT> typedef basic_string<_CharT> std::collate< _CharT >::string_type

Public typedefs.

Definition at line 591 of file locale_classes.h.

4.660.3 Constructor & Destructor Documentation

4.660.3.1 template<typename _CharT> std::collate< _CharT >::collate (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 611 of file locale_classes.h.

4.660.3.2 template<typename _CharT> std::collate< _CharT >::collate (__c_locale *__cloc*, size_t *__refs* = 0) [inline], [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__refs	Passed to the base facet class.

Definition at line 625 of file locale_classes.h.

4.660.3.3 template<typename _CharT> virtual std::collate< _CharT >::~collate ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 688 of file locale_classes.h.

4.660.4 Member Function Documentation

4.660.4.1 template<typename _CharT> int std::collate< _CharT >::compare (const _CharT * *__lo1*, const _CharT * *__hi1*, const _CharT * *__lo2*, const _CharT * *__hi2*) const [inline]

Compare two strings.

This function compares two strings and returns the result by calling collate::do_compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 642 of file locale_classes.h.

4.660.4.2 template<typename _CharT> int std::collate< _CharT >::do_compare (const _CharT * *__lo1*, const _CharT * *__hi1*, const _CharT * *__lo2*, const _CharT * *__hi2*) const [protected], [virtual]

Compare two strings.

This function is a hook for derived classes to change the value returned.

See Also

compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 161 of file locale_classes.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::length().

4.660.4.3 template<typename _CharT> long std::collate< _CharT >::do_hash (const _CharT * __lo, const _CharT * __hi) const [protected], [virtual]

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Hash value.

Definition at line 256 of file locale_classes.tcc.

4.660.4.4 template<typename _CharT> collate< _CharT >::string_type std::collate< _CharT >::do_transform (const _CharT * __lo, const _CharT * __hi) const [protected], [virtual]

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

__lo	Start.
__hi	End.

Returns

transformed string.

Definition at line 200 of file locale_classes.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Traits, _Alloc >::length(), and std::basic_string< _CharT, _Traits, _Alloc >::push_back().

4.660.4.5 template<typename _CharT> long **std::collate< _CharT >::hash** (const _CharT * *__lo*, const _CharT * *__hi*) const `[inline]`

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning collate::do_hash().

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Hash value.

Definition at line 675 of file locale_classes.h.

4.660.4.6 template<typename _CharT> **string_type std::collate< _CharT >::transform** (const _CharT * *__lo*, const _CharT * *__hi*) const `[inline]`

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning collate::do_transform().

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Transformed string_type.

Definition at line 661 of file locale_classes.h.

4.660.5 Member Data Documentation

4.660.5.1 template<typename _CharT> **locale::id std::collate< _CharT >::id** `[static]`

Numpunct facet id.

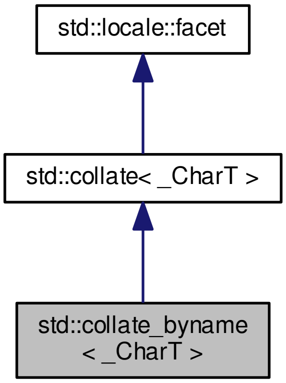
Definition at line 601 of file locale_classes.h.

The documentation for this class was generated from the following files:

- locale_classes.h
- locale_classes.tcc

4.661 std::collate_byname< _CharT > Class Template Reference

Inheritance diagram for std::collate_byname< _CharT >:



Public Types

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- **collate_byname** (const char ∗__s, size_t __refs=0)
- int **_M_compare** (const _CharT ∗, const _CharT ∗) const throw ()
- template<>
  int **_M_compare** (const char ∗, const char ∗) const throw()
- template<>
  int **_M_compare** (const wchar_t ∗, const wchar_t ∗) const throw()
- size_t **_M_transform** (_CharT ∗, const _CharT ∗, size_t) const throw ()
- template<>
  size_t **_M_transform** (char ∗, const char ∗, size_t) const throw()
- template<>
  size_t **_M_transform** (wchar_t ∗, const wchar_t ∗, size_t) const throw()
- int compare (const _CharT ∗__lo1, const _CharT ∗__hi1, const _CharT ∗__lo2, const _CharT ∗__hi2) const
- long hash (const _CharT ∗__lo, const _CharT ∗__hi) const
- string_type transform (const _CharT ∗__lo, const _CharT ∗__hi) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual int do_compare (const _CharT ∗__lo1, const _CharT ∗__hi1, const _CharT ∗__lo2, const _CharT ∗__hi2) const
- virtual long do_hash (const _CharT ∗__lo, const _CharT ∗__hi) const
- virtual string_type do_transform (const _CharT ∗__lo, const _CharT ∗__hi) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_collate**

4.661.1 Detailed Description

template<typename _CharT>class std::collate_byname< _CharT >

class collate_byname [22.2.4.2].

Definition at line 758 of file locale_classes.h.

4.661.2 Member Typedef Documentation

4.661.2.1 template<typename _CharT > typedef _CharT std::collate_byname< _CharT >::char_type

Public typedefs.

Definition at line 763 of file locale_classes.h.

4.661.2.2 template<typename _CharT > typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type

Public typedefs.

Definition at line 764 of file locale_classes.h.

4.661.3 Member Function Documentation

4.661.3.1 template<typename _CharT> int std::collate< _CharT >::compare (const _CharT ∗ *__lo1,* const _CharT ∗ *__hi1,* const _CharT ∗ *__lo2,* const _CharT ∗ *__hi2*) const [inline], [inherited]

Compare two strings.

This function compares two strings and returns the result by calling collate::do_compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 642 of file locale_classes.h.

4.661.3.2 template<typename _CharT> int std::collate< _CharT >::do_compare (const _CharT * *__lo1*, const _CharT * *__hi1*, const _CharT * *__lo2*, const _CharT * *__hi2*) const `[protected]`, `[virtual]`, `[inherited]`

Compare two strings.

This function is a hook for derived classes to change the value returned.

See Also

compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

Definition at line 161 of file locale_classes.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::length().

4.661.3.3 template<typename _CharT> long std::collate< _CharT >::do_hash (const _CharT * *__lo*, const _CharT * *__hi*) const `[protected]`, `[virtual]`, `[inherited]`

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Hash value.

Definition at line 256 of file locale_classes.tcc.

4.661.3.4 template<typename _CharT> collate< _CharT >::string_type std::collate< _CharT >::do_transform (const _CharT * __lo, const _CharT * __hi) const [protected], [virtual], [inherited]

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

Parameters

__lo	Start.
__hi	End.

Returns

transformed string.

Definition at line 200 of file locale_classes.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Traits, _Alloc >::length(), and std::basic_string< _CharT, _Traits, _Alloc >::push_back().

4.661.3.5 template<typename _CharT> long std::collate< _CharT >::hash (const _CharT * __lo, const _CharT * __hi) const [inline], [inherited]

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning collate::do_hash().

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Hash value.

Definition at line 675 of file locale_classes.h.

4.661.3.6 template<typename _CharT> string_type std::collate< _CharT >::transform (const _CharT * __lo, const _CharT * __hi) const [inline], [inherited]

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning collate::do_transform().

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Transformed string_type.

Definition at line 661 of file locale_classes.h.

4.661.4 Member Data Documentation

4.661.4.1 template<typename _CharT> locale::id std::collate< _CharT >::id [static], [inherited]

Numpunct facet id.

Definition at line 601 of file locale_classes.h.

The documentation for this class was generated from the following file:

- locale_classes.h

4.662 std::complex< _Tp > Struct Template Reference

Public Types

- typedef _Tp value_type

Public Member Functions

- constexpr complex (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())
- template<typename _Up >
  constexpr complex (const complex< _Up > &__z)
- **__attribute** ((__abi_tag__("cxx11"))) const expr _Tp real()
- **__attribute** ((__abi_tag__("cxx11"))) const expr _Tp imag()
- constexpr complex **__rep** () const
- void **imag** (_Tp __val)
- complex< _Tp > & operator*= (const _Tp &)
- template<typename _Up >
  complex< _Tp > & operator*= (const complex< _Up > &)
- complex< _Tp > & operator+= (const _Tp &__t)
- template<typename _Up >
  complex< _Tp > & operator+= (const complex< _Up > &)
- complex< _Tp > & operator-= (const _Tp &__t)
- template<typename _Up >
  complex< _Tp > & operator-= (const complex< _Up > &)
- complex< _Tp > & operator/= (const _Tp &)
- template<typename _Up >
  complex< _Tp > & operator/= (const complex< _Up > &)
- complex< _Tp > & operator= (const _Tp &)
- template<typename _Up >
  complex< _Tp > & operator= (const complex< _Up > &)
- void **real** (_Tp __val)

4.662.1 Detailed Description

template<typename _Tp>struct std::complex< _Tp >

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

Parameters

Tp	Type of real and imaginary values.

Definition at line 125 of file complex.

4.662.2 Member Typedef Documentation

4.662.2.1 template<typename _Tp> typedef _Tp std::complex< _Tp >::value_type

Value typedef.

Definition at line 128 of file complex.

4.662.3 Constructor & Destructor Documentation

4.662.3.1 template<typename _Tp> constexpr std::complex< _Tp >::complex (const _Tp & __r = `_Tp()`, const _Tp & __i = `_Tp()`) `[inline]`

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

Definition at line 132 of file complex.

4.662.3.2 template<typename _Tp> template<typename _Up > constexpr std::complex< _Tp >::complex (const complex< _Up > & __z) `[inline]`

Copy constructor.

Definition at line 139 of file complex.

4.662.4 Member Function Documentation

4.662.4.1 template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator+= (const _Tp & __t) `[inline]`

Add *t* to this complex number.

Definition at line 184 of file complex.

4.662.4.2 template<typename _Tp> complex<_Tp>& std::complex< _Tp >::operator-= (const _Tp & __t) `[inline]`

Subtract *t* from this complex number.

Definition at line 193 of file complex.

The documentation for this struct was generated from the following file:

- complex

4.663 std::complex< double > Struct Template Reference

Public Types

- typedef __complex__ double **_ComplexT**
- typedef double **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (double __r=0.0, double __i=0.0)
- **__attribute** ((__abi_tag__("cxx11"))) const expr double real()
- **__attribute** ((__abi_tag__("cxx11"))) const expr double imag()
- constexpr _ComplexT **__rep** () const
- void **imag** (double __val)
- complex & **operator*=** (double __d)
- template<typename _Tp >
  complex & **operator*=** (const complex< _Tp > &__z)
- complex & **operator+=** (double __d)
- template<typename _Tp >
  complex & **operator+=** (const complex< _Tp > &__z)
- complex & **operator-=** (double __d)
- template<typename _Tp >
  complex & **operator-=** (const complex< _Tp > &__z)
- complex & **operator/=** (double __d)
- template<typename _Tp >
  complex & **operator/=** (const complex< _Tp > &__z)
- complex & **operator=** (double __d)
- template<typename _Tp >
  complex & **operator=** (const complex< _Tp > &__z)
- void **real** (double __val)

4.663.1 Detailed Description

template<>struct std::complex< double >

26.2.3 complex specializations complex<double> specialization

Definition at line 1193 of file complex.

The documentation for this struct was generated from the following file:

- complex

4.664 std::complex< float > Struct Template Reference

Public Types

- typedef __complex__ float **_ComplexT**
- typedef float **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (float __r=0.0f, float __i=0.0f)
- constexpr **complex** (const complex< long double > &)
- **__attribute** ((__abi_tag__("cxx11"))) const expr float real()
- **__attribute** ((__abi_tag__("cxx11"))) const expr float imag()
- constexpr _ComplexT **__rep** () const
- void **imag** (float __val)
- complex & **operator∗=** (float __f)
- template<class _Tp >
  complex & **operator∗=** (const complex< _Tp > &__z)
- complex & **operator+=** (float __f)
- template<typename _Tp >
  complex & **operator+=** (const complex< _Tp > &__z)
- complex & **operator-=** (float __f)
- template<class _Tp >
  complex & **operator-=** (const complex< _Tp > &__z)
- complex & **operator/=** (float __f)
- template<class _Tp >
  complex & **operator/=** (const complex< _Tp > &__z)
- complex & **operator=** (float __f)
- template<typename _Tp >
  complex & **operator=** (const complex< _Tp > &__z)
- void **real** (float __val)

4.664.1 Detailed Description

template<>struct std::complex< float >

26.2.3 complex specializations complex<float> specialization

Definition at line 1044 of file complex.

The documentation for this struct was generated from the following file:

- complex

4.665 std::complex< long double > Struct Template Reference

Public Types

- typedef __complex__ long double **_ComplexT**
- typedef long double **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (long double __r=0.0L, long double __i=0.0L)

4.665.1 Detailed Description

template<>struct std::complex< long double >

26.2.3 complex specializations complex<long double> specialization

Definition at line 1343 of file complex.

The documentation for this struct was generated from the following file:

- complex

4.666 std::condition_variable Class Reference

Public Types

- typedef __native_type ∗ **native_handle_type**

Public Member Functions

- **condition_variable** (const condition_variable &)=delete
- native_handle_type **native_handle** ()
- void **notify_all** () noexcept
- void **notify_one** () noexcept
- condition_variable & **operator=** (const condition_variable &)=delete
- void **wait** (unique_lock< mutex > &__lock)
- template<typename _Predicate >
  void **wait** (unique_lock< mutex > &__lock, _Predicate __p)
- template<typename _Rep , typename _Period >
  cv_status **wait_for** (unique_lock< mutex > &__lock, const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Rep , typename _Period , typename _Predicate >
  bool **wait_for** (unique_lock< mutex > &__lock, const chrono::duration< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Duration >
  cv_status **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< __clock_t, _Duration > &__atime)
- template<typename _Clock , typename _Duration >
  cv_status **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< _Clock, _Duration > &__atime)
- template<typename _Clock , typename _Duration , typename _Predicate >
  bool **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< _Clock, _Duration > &__atime, _Predicate __p)

4.666.1 Detailed Description

condition_variable

Definition at line 59 of file condition_variable.

The documentation for this class was generated from the following file:

- condition_variable

4.667 std::condition_variable_any Class Reference

Public Member Functions

- **condition_variable_any** (const condition_variable_any &)=delete
- void **notify_all** () noexcept
- void **notify_one** () noexcept
- condition_variable_any & **operator=** (const condition_variable_any &)=delete
- template<typename _Lock >
  void **wait** (_Lock &__lock)
- template<typename _Lock , typename _Predicate >
  void **wait** (_Lock &__lock, _Predicate __p)
- template<typename _Lock , typename _Rep , typename _Period >
  cv_status **wait_for** (_Lock &__lock, const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Lock , typename _Rep , typename _Period , typename _Predicate >
  bool **wait_for** (_Lock &__lock, const chrono::duration< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Lock , typename _Clock , typename _Duration >
  cv_status **wait_until** (_Lock &__lock, const chrono::time_point< _Clock, _Duration > &__atime)
- template<typename _Lock , typename _Clock , typename _Duration , typename _Predicate >
  bool **wait_until** (_Lock &__lock, const chrono::time_point< _Clock, _Duration > &__atime, _Predicate __p)

4.667.1 Detailed Description

condition_variable_any

Definition at line 170 of file condition_variable.

The documentation for this class was generated from the following file:

- condition_variable

4.668 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference

Public Types

- typedef _Iftrue **type**

4.668.1 Detailed Description

template<bool _Cond, typename _Iftrue, typename _Iffalse>struct std::conditional< _Cond, _Iftrue, _Iffalse >

Define a member typedef `type` to one of two argument types.

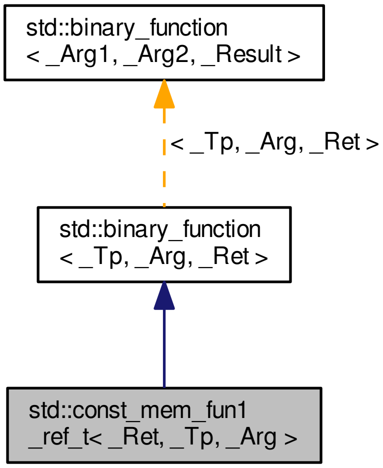
Definition at line 1780 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.669 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

Inheritance diagram for std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **const_mem_fun1_ref_t** (_Ret(_Tp::∗__pf)(_Arg) const)
- _Ret **operator()** (const _Tp &__r, _Arg __x) const

4.669.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

Definition at line 668 of file stl_function.h.

4.669.2 Member Typedef Documentation

4.669.2.1 typedef _Tp std::binary_function< _Tp , _Arg , _Ret >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.669.2.2 typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.669.2.3 typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

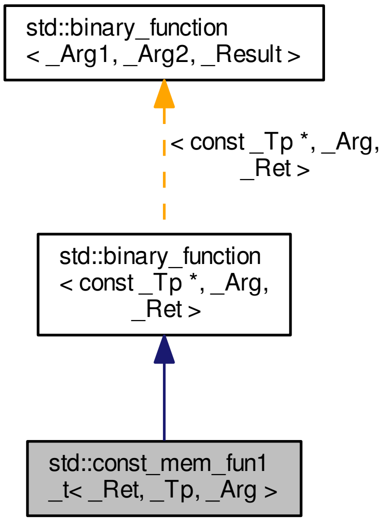
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.670 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

Inheritance diagram for std::const_mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef const _Tp ∗ first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **const_mem_fun1_t** (_Ret(_Tp::∗__pf)(_Arg) const)

- _Ret **operator()** (const _Tp ∗__p, _Arg __x) const

4.670.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>class std::const_mem_fun1_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

Definition at line 632 of file stl_function.h.

4.670.2 Member Typedef Documentation

4.670.2.1 typedef const _Tp ∗ **std::binary_function**< const _Tp ∗ , _Arg , _Ret >::**first_argument_type** `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.670.2.2 typedef _Ret **std::binary_function**< const _Tp ∗ , _Arg , _Ret >::**result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.670.2.3 typedef _Arg **std::binary_function**< const _Tp ∗ , _Arg , _Ret >::**second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

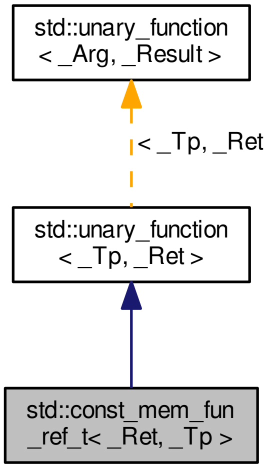
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.671 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference

Inheritance diagram for std::const_mem_fun_ref_t< _Ret, _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Ret result_type

Public Member Functions

- **const_mem_fun_ref_t** (_Ret(_Tp::∗__pf)() const)
- _Ret **operator()** (const _Tp &__r) const

4.671.1 Detailed Description

template<typename _Ret, typename _Tp>class std::const_mem_fun_ref_t< _Ret, _Tp >

One of the adaptors for member pointers.

Definition at line 596 of file stl_function.h.

4.671.2 Member Typedef Documentation

4.671.2.1 typedef _Tp std::unary_function< _Tp , _Ret >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.671.2.2 typedef _Ret std::unary_function< _Tp , _Ret >::result_type [inherited]

`result_type` is the return type

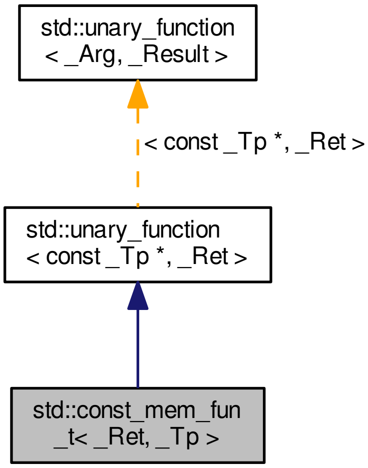
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.672 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference

Inheritance diagram for std::const_mem_fun_t< _Ret, _Tp >:



Public Types

- typedef const _Tp ∗ argument_type
- typedef _Ret result_type

Public Member Functions

- **const_mem_fun_t** (_Ret(_Tp::∗__pf)() const)
- _Ret **operator()** (const _Tp ∗__p) const

4.672.1 Detailed Description

template<typename _Ret, typename _Tp>class std::const_mem_fun_t< _Ret, _Tp >

One of the adaptors for member pointers.

Definition at line 560 of file stl_function.h.

4.672.2 Member Typedef Documentation

4.672.2.1 typedef const _Tp ∗ **std::unary_function**< const _Tp ∗ , _Ret >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.672.2.2 typedef _Ret **std::unary_function**< const _Tp ∗ , _Ret >**::result_type** `[inherited]`

`result_type` is the return type

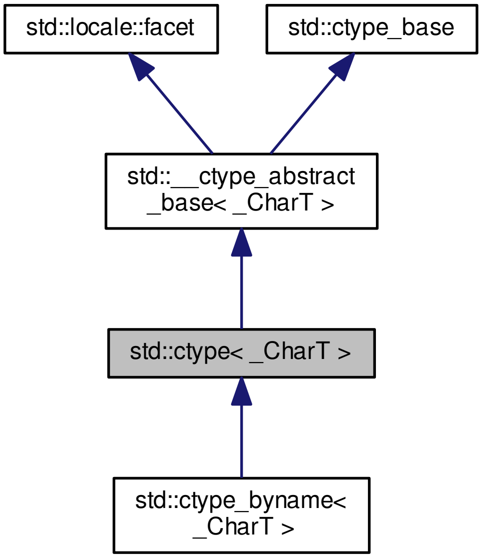
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.673 std::ctype< _CharT > Class Template Reference

Inheritance diagram for std::ctype< _CharT >:



Public Types

- typedef const int ∗ **__to_type**
- typedef _CharT **char_type**

- typedef __ctype_abstract_base
  < _CharT >::mask **mask**

Public Member Functions

- **ctype** (size_t __refs=0)
- bool is (mask __m, char_type __c) const
- const char_type ∗ is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- char narrow (char_type __c, char __dfault) const
- const char_type ∗ narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- const char_type ∗ scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- const char_type ∗ scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- char_type tolower (char_type __c) const
- const char_type ∗ tolower (char_type ∗__lo, const char_type ∗__hi) const
- char_type toupper (char_type __c) const
- const char_type ∗ toupper (char_type ∗__lo, const char_type ∗__hi) const
- char_type widen (char __c) const
- const char ∗ widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool do_is (mask __m, char_type __c) const
- virtual const char_type ∗ do_is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type ∗ do_narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- virtual const char_type ∗ do_scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- virtual const char_type ∗ do_scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type ∗ do_tolower (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char_type ∗ do_toupper (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_widen (char __c) const
- virtual const char ∗ do_widen (const char ∗__lo, const char ∗__hi, char_type ∗__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.673.1 Detailed Description

template<typename _CharT>class std::ctype< _CharT >

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in __ctype_abstract_base, to allow for implementation flexibility. See ctype<wchar_t> for an example. The functions are documented in __ctype_abstract_base.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 605 of file locale_facets.h.

4.673.2 Member Function Documentation

4.673.2.1 template<typename _CharT> virtual bool std::ctype< _CharT >::do_is (mask *__m*, char_type *__c*) const [protected], [virtual]

Test char_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.2 template<typename _CharT> virtual const char_type∗ std::ctype< _CharT >::do_is (const char_type ∗ *__lo*, const char_type ∗ *__hi*, mask ∗ *__vec*) const [protected], [virtual]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.3 template<typename _CharT> virtual char std::ctype< _CharT >::do_narrow (char_type *__c,* char *__dfault*) const [protected], [virtual]

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::narrow().

4.673.2.4 template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_narrow (const char_type * *__lo,* const char_type * *__hi,* char *__dfault,* char * *__to*) const [protected], [virtual]

Narrow char_type array to char.

This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.5 template<typename _CharT> virtual const **char_type**∗ **std::ctype**< _CharT >::do_scan_is (mask *__m,* const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi*) const [protected], [virtual]

Find char_type matching mask.

This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.6 template<typename _CharT> virtual const **char_type**∗ **std::ctype**< _CharT >::do_scan_not (mask *__m,* const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi*) const [protected], [virtual]

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.7 template<typename _CharT> virtual **char_type std::ctype**< _CharT >::do_tolower (**char_type** *__c*) const [protected], [virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::tolower().

4.673.2.8 template<typename _CharT> virtual const **char_type*** **std::ctype**< _CharT >::do_tolower (**char_type** * *__lo,* const **char_type** * *__hi*) const [protected], [virtual]

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.9 template<typename _CharT> virtual **char_type std::ctype**< _CharT >::do_toupper (**char_type** *__c*) const [protected], [virtual]

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::toupper().

4.673.2.10 template<typename _CharT> virtual const char_type∗ std::ctype< _CharT >::do_toupper (char_type ∗ *__lo*, const char_type ∗ *__hi*) const [protected], [virtual]

Convert array to uppercase.

This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.11 template<typename _CharT> virtual char_type std::ctype< _CharT >::do_widen (char *__c*) const [protected], [virtual]

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::widen().

4.673.2.12 template<typename _CharT> virtual const char∗ std::ctype< _CharT >::do_widen (const char ∗ *__lo,* const char ∗ *__hi,* char_type ∗ *__to*) const [protected], [virtual]

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.673.2.13 template<typename _CharT> bool std::__ctype_abstract_base< _CharT >::is (mask *__m,* char_type *__c*) const [inline], [inherited]

Test char_type classification.

This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__c	The char_type to compare the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Definition at line 162 of file locale_facets.h.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.673.2.14 template<typename _CharT> const char_type∗ std::__ctype_abstract_base< _CharT >::is (const char_type ∗ *__lo,* const char_type ∗ *__hi,* mask ∗ *__vec*) const [inline], [inherited]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__*hi*.

Definition at line 179 of file locale_facets.h.

4.673.2.15 template<typename _CharT> char std::__ctype_abstract_base< _CharT >::narrow (char_type *__c*, char *__dfault*) const [inline], [inherited]

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file locale_facets.h.

Referenced by std::time_put< _CharT, _OutIter >::put().

4.673.2.16 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::narrow (const char_type * *__lo*, const char_type * *__hi*, char *__dfault*, char * *__to*) const [inline], [inherited]

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__*hi*.

Definition at line 346 of file locale_facets.h.

4.673.2.17 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::scan_is (mask *__m*, const char_type * *__lo*, const char_type * *__hi*) const [inline], [inherited]

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

Definition at line 195 of file locale_facets.h.

4.673.2.18 template<typename _CharT> const **char_type*** **std::__ctype_abstract_base**< _CharT >::scan_not (mask *__m*, const **char_type** * *__lo*, const **char_type** * *__hi*) const [inline], [inherited]

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

__m	The mask to compare against.
__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else *__hi*.

Definition at line 211 of file locale_facets.h.

4.673.2.19 template<typename _CharT> **char_type std::__ctype_abstract_base**< _CharT >::tolower (**char_type** *__c*) const [inline], [inherited]

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Definition at line 254 of file locale_facets.h.

4.673.2.20 template<typename _CharT> const **char_type*** **std::__ctype_abstract_base**< _CharT >::tolower (**char_type** * *__lo*, const **char_type** * *__hi*) const [inline], [inherited]

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 269 of file locale_facets.h.

4.673.2.21 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::toupper (char_type *__c*) const [inline], [inherited]

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Definition at line 225 of file locale_facets.h.

4.673.2.22 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::toupper (char_type * *__lo*, const char_type * *__hi*) const [inline], [inherited]

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 240 of file locale_facets.h.

4.673.2.23 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::widen (char *__c*) const [inline], [inherited]

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_put< _CharT, _OutIter >::do_put(), std::money_put< _CharT, _OutIter >::do_put(), std::tr2::operator<<(), and std::operator<<().

4.673.2.24 template<typename _CharT> const char* **std::__ctype_abstract_base< _CharT >::widen** (const char * *__lo*, const char * *__hi*, **char_type** * *__to*) const [inline], [inherited]

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 305 of file locale_facets.h.

4.673.3 Member Data Documentation

4.673.3.1 template<typename _CharT> **locale::id std::ctype< _CharT >::id** [static]

The facet id for ctype<char_type>

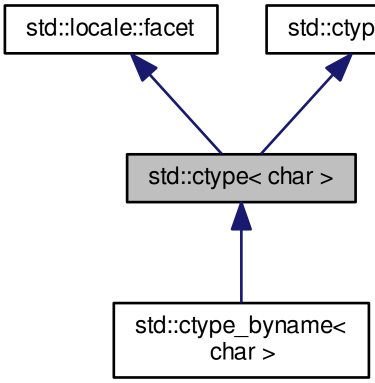
Definition at line 613 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.674 std::ctype< char > Class Template Reference

Inheritance diagram for std::ctype< char >:



Public Types

- typedef const int ∗ **__to_type**
- typedef char char_type
- typedef unsigned short **mask**

Public Member Functions

- ctype (const mask ∗__table=0, bool __del=false, size_t __refs=0)
- ctype (__c_locale __cloc, const mask ∗__table=0, bool __del=false, size_t __refs=0)
- bool is (mask __m, char __c) const
- const char ∗ is (const char ∗__lo, const char ∗__hi, mask ∗__vec) const
- char narrow (char_type __c, char __dfault) const
- const char_type ∗ narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- const char ∗ scan_is (mask __m, const char ∗__lo, const char ∗__hi) const
- const char ∗ scan_not (mask __m, const char ∗__lo, const char ∗__hi) const
- const mask ∗ table () const throw ()
- char_type tolower (char_type __c) const
- const char_type ∗ tolower (char_type ∗__lo, const char_type ∗__hi) const
- char_type toupper (char_type __c) const
- const char_type ∗ toupper (char_type ∗__lo, const char_type ∗__hi) const
- char_type widen (char __c) const
- const char ∗ widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Public Member Functions

- static const mask ∗ classic_table () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t table_size
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual ∼ctype ()
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type ∗ do_narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type ∗ do_tolower (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char_type ∗ do_toupper (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_widen (char __c) const
- virtual const char ∗ do_widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask ∗ **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

4.674.1 Detailed Description

template<>class std::ctype< char >

The ctype<char> specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 674 of file locale_facets.h.

4.674.2 Member Typedef Documentation

4.674.2.1 typedef char std::ctype< char >::char_type

Typedef for the template parameter char.

Definition at line 679 of file locale_facets.h.

4.674.3 Constructor & Destructor Documentation

4.674.3.1 std::ctype< char >::ctype (const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__table	If non-zero, table is used as the per-char mask. Else classic_table() is used.
__del	If true, passes ownership of table to this facet.
__refs	Passed to the base facet class.

4.674.3.2 std::ctype< char >::ctype (__c_locale __cloc, const mask * __table = 0, bool __del = false, size_t __refs = 0) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

__cloc	Handle to C locale data.
__table	If non-zero, table is used as the per-char mask.
__del	If true, passes ownership of table to this facet.
__refs	Passed to the base facet class.

4.674.3.3 virtual std::ctype< char >::~ctype ( ) [protected], [virtual]

Destructor.

This function deletes table() if *del* was true in the constructor.

4.674.4 Member Function Documentation

4.674.4.1 static const mask∗ std::ctype< char >::classic_table ( ) throw () [static]

Returns a pointer to the C locale mask table.

4.674.4.2 virtual char std::ctype< char >::do_narrow (char_type __c, char __dfault) const [inline], [protected], [virtual]

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Definition at line 1124 of file locale_facets.h.

4.674.4.3 virtual const char_type∗ std::ctype< char >::do_narrow (const char_type ∗ __lo, const char_type ∗ __hi, char __dfault, char ∗ __to) const [inline], [protected], [virtual]

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 1150 of file locale_facets.h.

4.674.4.4 virtual char_type std::ctype< char >::do_tolower (char_type __c) const [protected], [virtual]

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

4.674.4.5 virtual const **char_type**∗ **std::ctype**< char >::do_tolower (**char_type** ∗ *__lo*, const **char_type** ∗ *__hi*) const [protected], [virtual]

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

4.674.4.6 virtual **char_type** **std::ctype**< char >::do_toupper (**char_type** *__c*) const [protected], [virtual]

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The uppercase char if convertible, else *__c*.

4.674.4.7 virtual const **char_type**∗ **std::ctype**< char >::do_toupper (**char_type** ∗ *__lo*, const **char_type** ∗ *__hi*) const [protected], [virtual]

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

4.674.4.8 virtual char_type std::ctype< char >::do_widen (char *__c*) const [inline], [protected], [virtual]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted character.

Definition at line 1075 of file locale_facets.h.

4.674.4.9 virtual const char* std::ctype< char >::do_widen (const char * *__lo*, const char * *__hi*, char_type * *__to*) const [inline], [protected], [virtual]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 1098 of file locale_facets.h.

4.674.4.10 bool std::ctype< char >::is (mask *__m*, char *__c*) const [inline]

Test char classification.

This function compares the mask table[c] to *__m*.

Parameters

__c	The char to compare the mask of.
__m	The mask to compare against.

Returns

True if __m & table[__c] is true, false otherwise.

Definition at line 43 of file ctype_inline.h.

4.674.4.11 const char * std::ctype< char >::is (const char * *__lo*, const char * *__hi*, mask * *__vec*) const [inline]

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Definition at line 48 of file ctype_inline.h.

4.674.4.12 char std::ctype< char >::narrow (char_type *__c*, char *__dfault*) const [inline]

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

This function works as if it returns ctype<char>::do_narrow(c). do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted character.

Definition at line 923 of file locale_facets.h.

References std::ctype< _CharT >::do_narrow().

4.674.4.13 const char_type∗ std::ctype< char >::narrow (const char_type ∗ __lo, const char_type ∗ __hi, char __dfault, char ∗ __to) const [inline]

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do_narrow(lo, hi, dfault, to). do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 956 of file locale_facets.h.

References std::ctype< _CharT >::do_narrow().

4.674.4.14 const char ∗ std::ctype< char >::scan_is (mask __m, const char ∗ __lo, const char ∗ __hi) const [inline]

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char if found, else *__hi*.

Definition at line 57 of file ctype_inline.h.

4.674.4.15 const char ∗ std::ctype< char >::scan_not (mask __m, const char ∗ __lo, const char ∗ __hi) const [inline]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [__lo,__hi) for which is(m,char) is false.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *__hi*.

Definition at line 67 of file ctype_inline.h.

4.674.4.16 const mask∗ std::ctype< char >::table ( ) const throw () [inline]

Returns a pointer to the mask table provided to the constructor, or the default from classic_table() if none was provided.

Definition at line 974 of file locale_facets.h.

4.674.4.17 char_type std::ctype< char >::tolower (char_type *__c*) const [inline]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do_tolower(__c). do_tolower() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

Definition at line 828 of file locale_facets.h.

References std::ctype< _CharT >::do_tolower().

4.674.4.18 const char_type∗ std::ctype< char >::tolower (char_type ∗ *__lo*, const char_type ∗ *__hi*) const [inline]

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

tolower() acts as if it returns ctype<char>:: do_tolower(__lo, __hi). do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 845 of file locale_facets.h.

References std::ctype< _CharT >::do_tolower().

4.674.4.19 char_type std::ctype< char >::toupper (char_type __c) const [inline]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do_toupper(c). do_toupper() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The uppercase char if convertible, else *__c*.

Definition at line 795 of file locale_facets.h.

References std::ctype< _CharT >::do_toupper().

4.674.4.20 const char_type∗ std::ctype< char >::toupper (char_type ∗ __lo, const char_type ∗ __hi) const [inline]

Convert array to uppercase.

This function converts each char in the range [__lo,__hi) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>:: do_toupper(__lo, __hi). do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 812 of file locale_facets.h.

References std::ctype< _CharT >::do_toupper().

4.674.4.21 char_type std::ctype< char >::widen (char __c) const [inline]

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted character.

Definition at line 865 of file locale_facets.h.

References std::ctype< _CharT >::do_widen().

4.674.4.22 const char∗ std::ctype< char >::widen (const char ∗ __lo, const char ∗ __hi, char_type ∗ __to) const [inline]

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 892 of file locale_facets.h.

References std::ctype< _CharT >::do_widen().

4.674.5 Member Data Documentation

4.674.5.1 locale::id std::ctype< char >::id [static]

The facet id for ctype<char>

Definition at line 696 of file locale_facets.h.

4.674.5.2 const size_t std::ctype< char >::table_size [static]

The size of the mask table. It is SCHAR_MAX + 1.

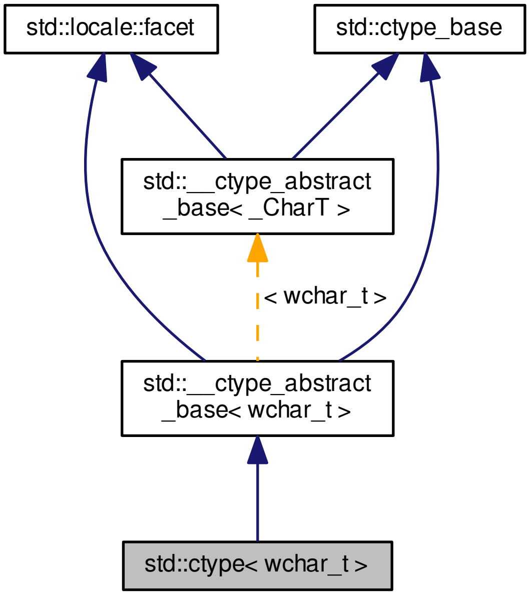
Definition at line 698 of file locale_facets.h.

The documentation for this class was generated from the following files:

- locale_facets.h
- ctype_inline.h

4.675 std::ctype< wchar_t > Class Template Reference

Inheritance diagram for std::ctype< wchar_t >:



Public Types

- typedef const int ∗ **__to_type**
- typedef wctype_t **__wmask_type**
- typedef wchar_t char_type
- typedef unsigned short **mask**

Public Member Functions

- ctype (size_t __refs=0)
- ctype (__c_locale __cloc, size_t __refs=0)
- bool is (mask __m, char_type __c) const
- const char_type ∗ is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- char narrow (char_type __c, char __dfault) const
- const char_type ∗ narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- const char_type ∗ scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- const char_type ∗ scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- char_type tolower (char_type __c) const
- const char_type ∗ tolower (char_type ∗__lo, const char_type ∗__hi) const
- char_type toupper (char_type __c) const
- const char_type ∗ toupper (char_type ∗__lo, const char_type ∗__hi) const

- char_type widen (char __c) const
- const char ∗ widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual ∼ctype ()
- __wmask_type **_M_convert_to_wmask** (const mask __m) const throw ()
- void **_M_initialize_ctype** () throw ()
- virtual bool do_is (mask __m, char_type __c) const
- virtual const char_type ∗ do_is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type ∗ do_narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- virtual const char_type ∗ do_scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- virtual const char_type ∗ do_scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type ∗ do_tolower (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char_type ∗ do_toupper (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_widen (char __c) const
- virtual const char ∗ do_widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- mask **_M_bit** [16]
- __c_locale **_M_c_locale_ctype**
- char **_M_narrow** [128]
- bool **_M_narrow_ok**
- wint_t **_M_widen** [1+static_cast< unsigned char >(-1)]
- __wmask_type **_M_wmask** [16]

4.675.1 Detailed Description

template<>class std::ctype< wchar_t >

The ctype<wchar_t> specialization.

This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.

ctype<wchar_t> inherits its public methods from __ctype_abstract_base<wchar_t>.

Definition at line 1175 of file locale_facets.h.

4.675.2 Member Typedef Documentation

4.675.2.1 typedef wchar_t std::ctype< wchar_t >::char_type

Typedef for the template parameter wchar_t.

Definition at line 1180 of file locale_facets.h.

4.675.3 Constructor & Destructor Documentation

4.675.3.1 std::ctype< wchar_t >::ctype (size_t *__refs* = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

4.675.3.2 std::ctype< wchar_t >::ctype (__c_locale *__cloc*, size_t *__refs* = 0) [explicit]

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

Parameters

__cloc	Handle to C locale data.
__refs	Passed to the base facet class.

4.675.3.3 virtual **std::ctype**< wchar_t >::∼**ctype** ( ) [protected], [virtual]

Destructor.

4.675.4 Member Function Documentation

4.675.4.1 virtual bool **std::ctype**< wchar_t >::do_is (mask *__m*, **char_type** *__c*) const [protected], [virtual]

Test wchar_t classification.

This function finds a mask M for *c* and compares it to mask *m*.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The wchar_t to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.2 virtual const **char_type**∗ **std::ctype**< wchar_t >::do_is (const **char_type** ∗ *__lo*, const **char_type** ∗ *__hi*, mask ∗ *__vec*) const [protected], [virtual]

Return a mask array.

This function finds the mask for each wchar_t in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.3 virtual char **std::ctype**< wchar_t >::do_narrow (**char_type** *__c*, char *__dfault*) const [protected], [virtual]

Narrow wchar_t to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<wchar_t> facet, *c* will be cast to char and returned.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The wchar_t to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.4 virtual const char_type* std::ctype< wchar_t >::do_narrow (const char_type * *__lo*, const char_type * *__hi*, char *__dfault*, char * *__to*) const [protected], [virtual]

Narrow wchar_t array to char array.

This virtual function converts each wchar_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar_t in the input that cannot be converted, *dfault* is used instead. For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to char.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.5 virtual const char_type* std::ctype< wchar_t >::do_scan_is (mask *__m*, const char_type * *__lo*, const char_type * *__hi*) const [protected], [virtual]

Find wchar_t matching mask.

This function searches for and returns the first wchar_t c in [__lo,__hi) for which is(__m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching wchar_t if found, else *__hi*.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.6 virtual const char_type* std::ctype< wchar_t >::do_scan_not (mask *__m*, const char_type * *__lo*, const char_type * *__hi*) const [protected], [virtual]

Find wchar_t not matching mask.

This function searches for and returns a pointer to the first wchar_t c of [__lo,__hi) for which is(__m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching wchar_t if found, else *__hi*.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.7 virtual char_type std::ctype< wchar_t >::do_tolower (char_type *__c*) const [protected], [virtual]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The wchar_t to convert.

Returns

The lowercase wchar_t if convertible, else *__c*.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.8 virtual const char_type* std::ctype< wchar_t >::do_tolower (char_type * *__lo*, const char_type * *__hi*) const [protected], [virtual]

Convert array to lowercase.

This virtual function converts each wchar_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.9 virtual char_type std::ctype< wchar_t >::do_toupper (char_type *__c*) const [protected], [virtual]

Convert to uppercase.

This virtual function converts the wchar_t argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The wchar_t to convert.

Returns

The uppercase wchar_t if convertible, else *__c*.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.10 virtual const char_type* std::ctype< wchar_t >::do_toupper (char_type * *__lo,* const char_type * *__hi*) const [protected], [virtual]

Convert array to uppercase.

This virtual function converts each wchar_t in the range [lo,hi) to uppercase if possible. Other elements remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.11 virtual char_type std::ctype< wchar_t >::do_widen (char *__c*) const [protected], [virtual]

Widen char to wchar_t.

This virtual function converts the char to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be cast to wchar_t.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted wchar_t.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.12 virtual const char∗ **std::ctype**< wchar_t >::do_widen (const char ∗ *__lo,* const char ∗ *__hi,* **char_type** ∗ *__to*) const [protected], [virtual]

Widen char array to wchar_t array.

This function converts each char in the input to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to wchar_t.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

4.675.4.13 bool **std::__ctype_abstract_base**< wchar_t >::is (mask *__m,* **char_type** *__c*) const [inline], [inherited]

Test char_type classification.

This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__c	The char_type to compare the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Definition at line 162 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_is().

4.675.4.14 const **char_type**∗ **std::__ctype_abstract_base**< wchar_t >::is (const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi,* mask ∗ *__vec*) const [inline], [inherited]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Definition at line 179 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_is().

4.675.4.15 char **std::__ctype_abstract_base**< wchar_t >::narrow (**char_type** *__c,* char *__dfault*) const [inline], [inherited]

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_narrow().

4.675.4.16 const **char_type**∗ **std::__ctype_abstract_base**< wchar_t >::narrow (const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi,* char *__dfault,* char ∗ *__to*) const [inline], [inherited]

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 346 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_narrow().

4.675.4.17 const **char_type**∗ **std::__ctype_abstract_base**< wchar_t >::scan_is (mask *__m*, const **char_type** ∗ *__lo*, const **char_type** ∗ *__hi*) const [inline], [inherited]

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

Definition at line 195 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_scan_is().

4.675.4.18 const **char_type**∗ **std::__ctype_abstract_base**< wchar_t >::scan_not (mask *__m*, const **char_type** ∗ *__lo*, const **char_type** ∗ *__hi*) const [inline], [inherited]

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

__m	The mask to compare against.
__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else *__hi*.

Definition at line 211 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_scan_not().

4.675.4.19 char_type std::__ctype_abstract_base< wchar_t >::tolower (char_type __c) const [inline], [inherited]

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Definition at line 254 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_tolower().

4.675.4.20 const char_type* std::__ctype_abstract_base< wchar_t >::tolower (char_type * __lo, const char_type * __hi) const [inline], [inherited]

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 269 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_tolower().

4.675.4.21 char_type std::__ctype_abstract_base< wchar_t >::toupper (char_type __c) const [inline], [inherited]

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Definition at line 225 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_toupper().

4.675.4.22 const char_type* std::__ctype_abstract_base< wchar_t >::toupper (char_type * __lo, const char_type * __hi) const [inline], [inherited]

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 240 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_toupper().

4.675.4.23 char_type std::__ctype_abstract_base< wchar_t >::widen (char __c) const [inline], [inherited]

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_widen().

4.675.4.24 const char* std::__ctype_abstract_base< wchar_t >::widen (const char * __lo, const char * __hi, char_type * __to) const [inline], [inherited]

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 305 of file locale_facets.h.

References std::__ctype_abstract_base< _CharT >::do_widen().

4.675.5 Member Data Documentation

4.675.5.1 locale::id std::ctype< wchar_t >::id [static]

The facet id for ctype<wchar_t>

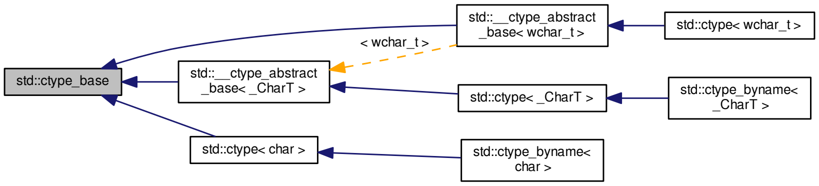
Definition at line 1198 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.676 std::ctype_base Struct Reference

Inheritance diagram for std::ctype_base:



Public Types

- typedef const int * **__to_type**
- typedef unsigned short **mask**

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

4.676.1 Detailed Description

Base class for ctype.

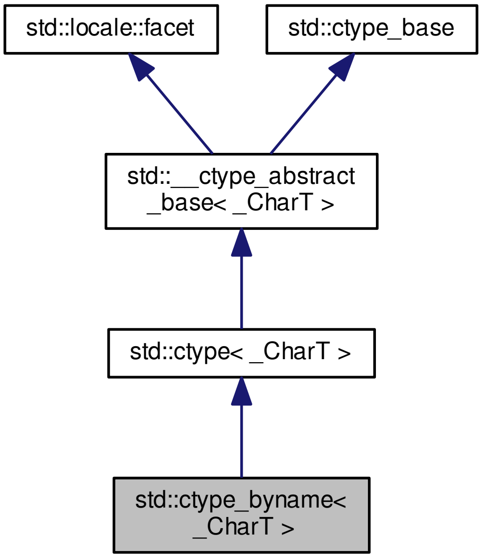
Definition at line 41 of file ctype_base.h.

The documentation for this struct was generated from the following file:

- ctype_base.h

4.677 std::ctype_byname< _CharT > Class Template Reference

Inheritance diagram for std::ctype_byname< _CharT >:



Public Types

- typedef const int ∗ **__to_type**
- typedef _CharT **char_type**
- typedef ctype< _CharT >::mask **mask**

Public Member Functions

- **ctype_byname** (const char ∗__s, size_t __refs=0)
- bool is (mask __m, char_type __c) const
- const char_type ∗ is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- char narrow (char_type __c, char __dfault) const

- const char_type ∗ narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- const char_type ∗ scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- const char_type ∗ scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- char_type tolower (char_type __c) const
- const char_type ∗ tolower (char_type ∗__lo, const char_type ∗__hi) const
- char_type toupper (char_type __c) const
- const char_type ∗ toupper (char_type ∗__lo, const char_type ∗__hi) const
- char_type widen (char __c) const
- const char ∗ widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool do_is (mask __m, char_type __c) const
- virtual const char_type ∗ do_is (const char_type ∗__lo, const char_type ∗__hi, mask ∗__vec) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type ∗ do_narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- virtual const char_type ∗ do_scan_is (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- virtual const char_type ∗ do_scan_not (mask __m, const char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type ∗ do_tolower (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char_type ∗ do_toupper (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_widen (char __c) const
- virtual const char ∗ do_widen (const char ∗__lo, const char ∗__hi, char_type ∗__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.677.1 Detailed Description

template<typename _CharT>class std::ctype_byname< _CharT >

class ctype_byname [22.2.1.2].

Definition at line 1467 of file locale_facets.h.

4.677.2 Member Function Documentation

4.677.2.1 template<typename _CharT> virtual bool std::ctype< _CharT >::do_is (mask *__m*, char_type *__c*) const [protected], [virtual], [inherited]

Test char_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.2 template<typename _CharT> virtual const char_type* std::ctype< _CharT >::do_is (const char_type * *__lo*, const char_type * *__hi*, mask * *__vec*) const [protected], [virtual], [inherited]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.3 template<typename _CharT> virtual char **std::ctype**< _CharT >::do_narrow (**char_type** *__c,* char *__dfault*) const `[protected]`, `[virtual]`, `[inherited]`

Narrow char_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::narrow().

4.677.2.4 template<typename _CharT> virtual const **char_type**∗ **std::ctype**< _CharT >::do_narrow (const **char_type** ∗ *__lo,* const **char_type** ∗ *__hi,* char *__dfault,* char ∗ *__to*) const `[protected]`, `[virtual]`, `[inherited]`

Narrow char_type array to char.

This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.5 template<typename _CharT> virtual const char_type∗ std::ctype< _CharT >::do_scan_is (mask __m, const char_type ∗ __lo, const char_type ∗ __hi) const [protected], [virtual], [inherited]

Find char_type matching mask.

This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.6 template<typename _CharT> virtual const char_type∗ std::ctype< _CharT >::do_scan_not (mask __m, const char_type ∗ __lo, const char_type ∗ __hi) const [protected], [virtual], [inherited]

Find char_type not matching mask.

This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.

do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.7 template<typename _CharT> virtual char_type std::ctype< _CharT >::do_tolower (char_type __c) const [protected], [virtual], [inherited]

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::tolower().

4.677.2.8 template<typename _CharT> virtual const **char_type**∗ **std::ctype**< _CharT >::do_tolower (**char_type** ∗ *__lo*, const **char_type** ∗ *__hi*) const [protected], [virtual], [inherited]

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.9 template<typename _CharT> virtual **char_type** **std::ctype**< _CharT >::do_toupper (**char_type** *__c*) const [protected], [virtual], [inherited]

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::toupper().

4.677.2.10 template<typename _CharT> virtual const **char_type**∗ **std::ctype**< _CharT >::do_toupper (**char_type** ∗ *__lo*, const **char_type** ∗ *__hi*) const [protected], [virtual], [inherited]

Convert array to uppercase.

This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.11 template<typename _CharT> virtual **char_type std::ctype**< _CharT >::do_widen (char *__c*) const [protected], [virtual], [inherited]

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::widen().

4.677.2.12 template<typename _CharT> virtual const char* **std::ctype**< _CharT >::do_widen (const char * *__lo,* const char * *__hi,* **char_type** * *__to*) const [protected], [virtual], [inherited]

Widen char array.

This function converts each char in the input to char_type using the simplest reasonable transformation.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

4.677.2.13 template<typename _CharT> bool std::__ctype_abstract_base< _CharT >::is (mask *__m*, char_type *__c*) const [inline], [inherited]

Test char_type classification.

This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__c	The char_type to compare the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Definition at line 162 of file locale_facets.h.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.677.2.14 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::is (const char_type * *__lo*, const char_type * *__hi*, mask * *__vec*) const [inline], [inherited]

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Definition at line 179 of file locale_facets.h.

4.677.2.15 template<typename _CharT> char std::__ctype_abstract_base< _CharT >::narrow (char_type *__c*, char *__dfault*) const [inline], [inherited]

Narrow char_type to char.

This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Definition at line 324 of file locale_facets.h.

Referenced by std::time_put< _CharT, _OutIter >::put().

4.677.2.16 template<typename _CharT> const char_type∗ std::__ctype_abstract_base< _CharT >::narrow (const char_type ∗ *__lo,* const char_type ∗ *__hi,* char *__dfault,* char ∗ *__to*) const [inline], [inherited]

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 346 of file locale_facets.h.

4.677.2.17 template<typename _CharT> const char_type∗ std::__ctype_abstract_base< _CharT >::scan_is (mask *__m,* const char_type ∗ *__lo,* const char_type ∗ *__hi*) const [inline], [inherited]

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

Definition at line 195 of file locale_facets.h.

4.677.2.18 template<typename _CharT> const char_type∗ std::__ctype_abstract_base< _CharT >::scan_not (mask *__m*, const char_type ∗ *__lo*, const char_type ∗ *__hi*) const [inline], [inherited]

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

__m	The mask to compare against.
__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else *__hi*.

Definition at line 211 of file locale_facets.h.

4.677.2.19 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::tolower (char_type *__c*) const [inline], [inherited]

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Definition at line 254 of file locale_facets.h.

4.677.2.20 template<typename _CharT> const char_type∗ std::__ctype_abstract_base< _CharT >::tolower (char_type ∗ *__lo*, const char_type ∗ *__hi*) const [inline], [inherited]

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 269 of file locale_facets.h.

4.677.2.21 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::toupper (char_type __c) const [inline], [inherited]

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Definition at line 225 of file locale_facets.h.

4.677.2.22 template<typename _CharT> const char_type* std::__ctype_abstract_base< _CharT >::toupper (char_type * __lo, const char_type * __hi) const [inline], [inherited]

Convert array to uppercase.

This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 240 of file locale_facets.h.

4.677.2.23 template<typename _CharT> char_type std::__ctype_abstract_base< _CharT >::widen (char __c) const [inline], [inherited]

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type.

Definition at line 286 of file locale_facets.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_put< _CharT, _OutIter >::do_put(), std::money_put< _CharT, _OutIter >::do_put(), std::tr2::operator<<(), and std::operator<<().

4.677.2.24 template<typename _CharT> const char* **std::__ctype_abstract_base< _CharT >::widen** (const char * *__lo,* const char * *__hi,* **char_type** * *__to*) const [inline], [inherited]

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 305 of file locale_facets.h.

4.677.3 Member Data Documentation

4.677.3.1 template<typename _CharT> **locale::id std::ctype< _CharT >::id** [static], [inherited]

The facet id for ctype<char_type>

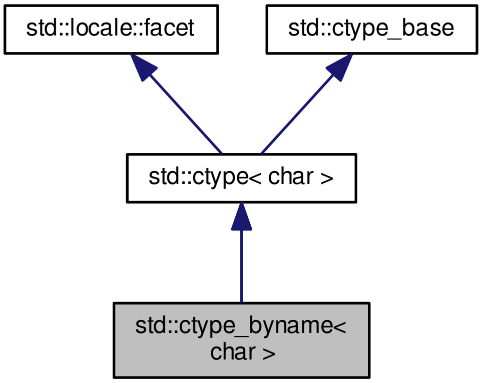
Definition at line 613 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.678 std::ctype_byname< char > Class Template Reference

Inheritance diagram for std::ctype_byname< char >:



Public Types

- typedef const int ∗ **__to_type**
- typedef char char_type
- typedef unsigned short **mask**

Public Member Functions

- **ctype_byname** (const char ∗__s, size_t __refs=0)
- bool is (mask __m, char __c) const
- const char ∗ is (const char ∗__lo, const char ∗__hi, mask ∗__vec) const
- char narrow (char_type __c, char __dfault) const
- const char_type ∗ narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- const char ∗ scan_is (mask __m, const char ∗__lo, const char ∗__hi) const
- const char ∗ scan_not (mask __m, const char ∗__lo, const char ∗__hi) const
- const mask ∗ table () const throw ()
- char_type tolower (char_type __c) const
- const char_type ∗ tolower (char_type ∗__lo, const char_type ∗__hi) const
- char_type toupper (char_type __c) const
- const char_type ∗ toupper (char_type ∗__lo, const char_type ∗__hi) const
- char_type widen (char __c) const
- const char ∗ widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Public Member Functions

- static const mask ∗ classic_table () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t table_size
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type ∗ do_narrow (const char_type ∗__lo, const char_type ∗__hi, char __dfault, char ∗__to) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type ∗ do_tolower (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char_type ∗ do_toupper (char_type ∗__lo, const char_type ∗__hi) const
- virtual char_type do_widen (char __c) const
- virtual const char ∗ do_widen (const char ∗__lo, const char ∗__hi, char_type ∗__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask ∗ **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

4.678.1 Detailed Description

template<>class std::ctype_byname< char >

22.2.1.4 Class ctype_byname specializations.

Definition at line 1482 of file locale_facets.h.

4.678.2 Member Typedef Documentation

4.678.2.1 typedef char std::ctype< char >::char_type `[inherited]`

Typedef for the template parameter char.

Definition at line 679 of file locale_facets.h.

4.678.3 Member Function Documentation

4.678.3.1 static const mask* std::ctype< char >::classic_table ( ) throw () `[static]`, `[inherited]`

Returns a pointer to the C locale mask table.

4.678.3.2 virtual char std::ctype< char >::do_narrow (char_type *__c*, char *__dfault*) const `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Definition at line 1124 of file locale_facets.h.

4.678.3.3 virtual const char_type* std::ctype< char >::do_narrow (const char_type * *__lo*, const char_type * *__hi*, char *__dfault*, char * *__to*) const `[inline]`, `[protected]`, `[virtual]`, `[inherited]`

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 1150 of file locale_facets.h.

4.678.3.4 virtual char_type std::ctype< char >::do_tolower (char_type *__c*) const [protected], [virtual], [inherited]

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

4.678.3.5 virtual const char_type* std::ctype< char >::do_tolower (char_type * *__lo,* const char_type * *__hi*) const [protected], [virtual], [inherited]

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

4.678.3.6 virtual char_type std::ctype< char >::do_toupper (char_type *__c*) const [protected], [virtual], [inherited]

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The uppercase char if convertible, else *__c*.

4.678.3.7 virtual const char_type∗ std::ctype< char >::do_toupper (char_type ∗ *__lo*, const char_type ∗ *__hi*) const [protected], [virtual], [inherited]

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

4.678.3.8 virtual char_type std::ctype< char >::do_widen (char *__c*) const [inline], [protected], [virtual], [inherited]

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted character.

Definition at line 1075 of file locale_facets.h.

4.678.3.9 virtual const char∗ std::ctype< char >::do_widen (const char ∗ __lo, const char ∗ __hi, char_type ∗ __to) const [inline], [protected], [virtual], [inherited]

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 1098 of file locale_facets.h.

4.678.3.10 bool std::ctype< char >::is (mask __m, char __c) const [inline], [inherited]

Test char classification.

This function compares the mask table[c] to *__m*.

Parameters

__c	The char to compare the mask of.
__m	The mask to compare against.

Returns

True if __m & table[__c] is true, false otherwise.

Definition at line 43 of file ctype_inline.h.

4.678.3.11 const char ∗ std::ctype< char >::is (const char ∗ __lo, const char ∗ __hi, mask ∗ __vec) const [inline], [inherited]

Return a mask array.

This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Definition at line 48 of file ctype_inline.h.

4.678.3.12 char std::ctype< char >::narrow (char_type *__c*, char *__dfault*) const [inline], [inherited]

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

This function works as if it returns ctype<char>::do_narrow(c). do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted character.

Definition at line 923 of file locale_facets.h.

References std::ctype< _CharT >::do_narrow().

4.678.3.13 const char_type∗ std::ctype< char >::narrow (const char_type ∗ *__lo*, const char_type ∗ *__hi*, char *__dfault*, char ∗ *__to*) const [inline], [inherited]

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do_narrow(lo, hi, dfault, to). do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 956 of file locale_facets.h.

References std::ctype< _CharT >::do_narrow().

4.678.3.14 const char * **std::ctype**< char >::scan_is (mask *__m*, const char * *__lo*, const char * *__hi*) const [inline], [inherited]

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char if found, else *__hi*.

Definition at line 57 of file ctype_inline.h.

4.678.3.15 const char * **std::ctype**< char >::scan_not (mask *__m*, const char * *__lo*, const char * *__hi*) const [inline], [inherited]

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [__lo,__hi) for which is(m,char) is false.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *__hi*.

Definition at line 67 of file ctype_inline.h.

4.678.3.16 const mask* **std::ctype**< char >::table ( ) const throw () [inline], [inherited]

Returns a pointer to the mask table provided to the constructor, or the default from classic_table() if none was provided.

Definition at line 974 of file locale_facets.h.

4.678.3.17 char_type **std::ctype**< char >::tolower (char_type *__c*) const [inline], [inherited]

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do_tolower(__c). do_tolower() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

Definition at line 828 of file locale_facets.h.

References std::ctype< _CharT >::do_tolower().

4.678.3.18 const char_type∗ std::ctype< char >::tolower (char_type ∗ *__lo,* const char_type ∗ *__hi*) const [inline], [inherited]

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

tolower() acts as if it returns ctype<char>:: do_tolower(__lo, __hi). do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 845 of file locale_facets.h.

References std::ctype< _CharT >::do_tolower().

4.678.3.19 char_type std::ctype< char >::toupper (char_type *__c*) const [inline], [inherited]

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do_toupper(c). do_toupper() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The uppercase char if convertible, else *__c*.

Definition at line 795 of file locale_facets.h.

References std::ctype< _CharT >::do_toupper().

4.678.3.20 const char_type∗ std::ctype< char >::toupper (char_type ∗ *__lo,* const char_type ∗ *__hi*) const [inline], [inherited]

Convert array to uppercase.

This function converts each char in the range [__lo,__hi) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>:: do_toupper(__lo, __hi). do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

Definition at line 812 of file locale_facets.h.

References std::ctype< _CharT >::do_toupper().

4.678.3.21 char_type std::ctype< char >::widen (char *__c*) const [inline], [inherited]

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted character.

Definition at line 865 of file locale_facets.h.

References std::ctype< _CharT >::do_widen().

4.678.3.22 const char* std::ctype< char >::widen (const char * *__lo,* const char * *__hi,* char_type * *__to*) const [inline], [inherited]

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Definition at line 892 of file locale_facets.h.

References std::ctype< _CharT >::do_widen().

4.678.4 Member Data Documentation

4.678.4.1 locale::id std::ctype< char >::id [static], [inherited]

The facet id for ctype<char>

Definition at line 696 of file locale_facets.h.

4.678.4.2 const size_t std::ctype< char >::table_size [static], [inherited]

The size of the mask table. It is SCHAR_MAX + 1.

Definition at line 698 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.679 std::decay< _Tp > Class Template Reference

Public Types

- typedef __decay_selector < __remove_type >::__type **type**

4.679.1 Detailed Description

template<typename _Tp>class std::decay< _Tp >

decay

Definition at line 1725 of file type_traits.

The documentation for this class was generated from the following file:

- type_traits

4.680 std::decimal::decimal128 Class Reference

Public Types

- typedef float __decfloat128 **__attribute__** ((mode(TD)))

Public Member Functions

- **decimal128** (decimal32 d32)

- **decimal128** (decimal64 d64)
- **decimal128** (float __r)
- **decimal128** (double __r)
- **decimal128** (long double __r)
- **decimal128** (int __z)
- **decimal128** (unsigned int __z)
- **decimal128** (long __z)
- **decimal128** (unsigned long __z)
- **decimal128** (long long __z)
- **decimal128** (unsigned long long __z)
- decimal128 (__decfloat128 __z)
- __decfloat128 **__getval** (void)
- void **__setval** (__decfloat128 __x)
- decimal128 & **operator∗=** (decimal32 __rhs)
- decimal128 & **operator∗=** (decimal64 __rhs)
- decimal128 & **operator∗=** (decimal128 __rhs)
- decimal128 & **operator∗=** (int __rhs)
- decimal128 & **operator∗=** (unsigned int __rhs)
- decimal128 & **operator∗=** (long __rhs)
- decimal128 & **operator∗=** (unsigned long __rhs)
- decimal128 & **operator∗=** (unsigned long long __rhs)
- decimal128 & **operator∗=** (long long __rhs)
- decimal128 & **operator++** ()
- decimal128 **operator++** (int)
- decimal128 & **operator+=** (int __rhs)
- decimal128 & **operator+=** (decimal32 __rhs)
- decimal128 & **operator+=** (decimal64 __rhs)
- decimal128 & **operator+=** (decimal128 __rhs)
- decimal128 & **operator+=** (unsigned int __rhs)
- decimal128 & **operator+=** (long __rhs)
- decimal128 & **operator+=** (unsigned long __rhs)
- decimal128 & **operator+=** (long long __rhs)
- decimal128 & **operator+=** (unsigned long long __rhs)
- decimal128 & **operator--** ()
- decimal128 **operator--** (int)
- decimal128 & **operator-=** (long __rhs)
- decimal128 & **operator-=** (int __rhs)
- decimal128 & **operator-=** (long long __rhs)
- decimal128 & **operator-=** (unsigned long long __rhs)
- decimal128 & **operator-=** (unsigned long __rhs)
- decimal128 & **operator-=** (decimal128 __rhs)
- decimal128 & **operator-=** (unsigned int __rhs)
- decimal128 & **operator-=** (decimal32 __rhs)
- decimal128 & **operator-=** (decimal64 __rhs)
- decimal128 & **operator/=** (decimal32 __rhs)
- decimal128 & **operator/=** (long long __rhs)
- decimal128 & **operator/=** (unsigned long long __rhs)
- decimal128 & **operator/=** (unsigned long __rhs)
- decimal128 & **operator/=** (decimal64 __rhs)
- decimal128 & **operator/=** (unsigned int __rhs)
- decimal128 & **operator/=** (decimal128 __rhs)
- decimal128 & **operator/=** (long __rhs)
- decimal128 & **operator/=** (int __rhs)

4.680.1 Detailed Description

3.2.4 Class decimal128.

Definition at line 393 of file decimal.

4.680.2 Constructor & Destructor Documentation

4.680.2.1 std::decimal::decimal128::decimal128 (__decfloat128 *__z*) `[inline]`

Conforming extension: Conversion from scalar decimal type.

Definition at line 418 of file decimal.

The documentation for this class was generated from the following file:

- decimal

4.681 std::decimal::decimal32 Class Reference

Public Types

- typedef float __decfloat32 **__attribute__** ((mode(SD)))

Public Member Functions

- **decimal32** (decimal64 __d64)
- **decimal32** (decimal128 __d128)
- **decimal32** (float __r)
- **decimal32** (double __r)
- **decimal32** (long double __r)
- **decimal32** (int __z)
- **decimal32** (unsigned int __z)
- **decimal32** (long __z)
- **decimal32** (unsigned long __z)
- **decimal32** (long long __z)
- **decimal32** (unsigned long long __z)
- decimal32 (__decfloat32 __z)
- __decfloat32 **__getval** (void)
- void **__setval** (__decfloat32 __x)
- decimal32 & **operator∗=** (decimal32 __rhs)
- decimal32 & **operator∗=** (decimal64 __rhs)
- decimal32 & **operator∗=** (decimal128 __rhs)
- decimal32 & **operator∗=** (int __rhs)
- decimal32 & **operator∗=** (unsigned int __rhs)
- decimal32 & **operator∗=** (long __rhs)
- decimal32 & **operator∗=** (unsigned long __rhs)
- decimal32 & **operator∗=** (unsigned long long __rhs)
- decimal32 & **operator∗=** (long long __rhs)
- decimal32 & **operator++** ()
- decimal32 **operator++** (int)
- decimal32 & **operator+=** (int __rhs)

- decimal32 & **operator+=** (decimal32 __rhs)
- decimal32 & **operator+=** (decimal64 __rhs)
- decimal32 & **operator+=** (decimal128 __rhs)
- decimal32 & **operator+=** (unsigned int __rhs)
- decimal32 & **operator+=** (long __rhs)
- decimal32 & **operator+=** (unsigned long __rhs)
- decimal32 & **operator+=** (long long __rhs)
- decimal32 & **operator+=** (unsigned long long __rhs)
- decimal32 & **operator--** ()
- decimal32 **operator--** (int)
- decimal32 & **operator-=** (long __rhs)
- decimal32 & **operator-=** (int __rhs)
- decimal32 & **operator-=** (long long __rhs)
- decimal32 & **operator-=** (unsigned long long __rhs)
- decimal32 & **operator-=** (unsigned long __rhs)
- decimal32 & **operator-=** (decimal128 __rhs)
- decimal32 & **operator-=** (unsigned int __rhs)
- decimal32 & **operator-=** (decimal32 __rhs)
- decimal32 & **operator-=** (decimal64 __rhs)
- decimal32 & **operator/=** (decimal32 __rhs)
- decimal32 & **operator/=** (long long __rhs)
- decimal32 & **operator/=** (unsigned long long __rhs)
- decimal32 & **operator/=** (unsigned long __rhs)
- decimal32 & **operator/=** (decimal64 __rhs)
- decimal32 & **operator/=** (unsigned int __rhs)
- decimal32 & **operator/=** (decimal128 __rhs)
- decimal32 & **operator/=** (long __rhs)
- decimal32 & **operator/=** (int __rhs)

4.681.1 Detailed Description

3.2.2 Class decimal32.

Definition at line 227 of file decimal.

4.681.2 Constructor & Destructor Documentation

4.681.2.1 std::decimal::decimal32::decimal32 (__decfloat32 *__z*) [inline]

Conforming extension: Conversion from scalar decimal type.

Definition at line 251 of file decimal.

The documentation for this class was generated from the following file:

- decimal

4.682 std::decimal::decimal64 Class Reference

Public Types

- typedef float __decfloat64 **__attribute__** ((mode(DD)))

Public Member Functions

- **decimal64** (decimal32 d32)
- **decimal64** (decimal128 d128)
- **decimal64** (float __r)
- **decimal64** (double __r)
- **decimal64** (long double __r)
- **decimal64** (int __z)
- **decimal64** (unsigned int __z)
- **decimal64** (long __z)
- **decimal64** (unsigned long __z)
- **decimal64** (long long __z)
- **decimal64** (unsigned long long __z)
- decimal64 (__decfloat64 __z)
- __decfloat64 **__getval** (void)
- void **__setval** (__decfloat64 __x)
- decimal64 & **operator*=** (decimal32 __rhs)
- decimal64 & **operator*=** (decimal64 __rhs)
- decimal64 & **operator*=** (decimal128 __rhs)
- decimal64 & **operator*=** (int __rhs)
- decimal64 & **operator*=** (unsigned int __rhs)
- decimal64 & **operator*=** (long __rhs)
- decimal64 & **operator*=** (unsigned long __rhs)
- decimal64 & **operator*=** (unsigned long long __rhs)
- decimal64 & **operator*=** (long long __rhs)
- decimal64 & **operator++** ()
- decimal64 **operator++** (int)
- decimal64 & **operator+=** (int __rhs)
- decimal64 & **operator+=** (decimal32 __rhs)
- decimal64 & **operator+=** (decimal64 __rhs)
- decimal64 & **operator+=** (decimal128 __rhs)
- decimal64 & **operator+=** (unsigned int __rhs)
- decimal64 & **operator+=** (long __rhs)
- decimal64 & **operator+=** (unsigned long __rhs)
- decimal64 & **operator+=** (long long __rhs)
- decimal64 & **operator+=** (unsigned long long __rhs)
- decimal64 & **operator--** ()
- decimal64 **operator--** (int)
- decimal64 & **operator-=** (long __rhs)
- decimal64 & **operator-=** (int __rhs)
- decimal64 & **operator-=** (long long __rhs)
- decimal64 & **operator-=** (unsigned long long __rhs)
- decimal64 & **operator-=** (unsigned long __rhs)
- decimal64 & **operator-=** (decimal128 __rhs)
- decimal64 & **operator-=** (unsigned int __rhs)
- decimal64 & **operator-=** (decimal32 __rhs)
- decimal64 & **operator-=** (decimal64 __rhs)
- decimal64 & **operator/=** (decimal32 __rhs)
- decimal64 & **operator/=** (long long __rhs)
- decimal64 & **operator/=** (unsigned long long __rhs)
- decimal64 & **operator/=** (unsigned long __rhs)

- decimal64 & **operator/=** (decimal64 __rhs)
- decimal64 & **operator/=** (unsigned int __rhs)
- decimal64 & **operator/=** (decimal128 __rhs)
- decimal64 & **operator/=** (long __rhs)
- decimal64 & **operator/=** (int __rhs)

4.682.1 Detailed Description

3.2.3 Class decimal64.

Definition at line 310 of file decimal.

4.682.2 Constructor & Destructor Documentation

4.682.2.1 std::decimal::decimal64::decimal64 (__decfloat64 *__z*) [inline]

Conforming extension: Conversion from scalar decimal type.

Definition at line 334 of file decimal.

The documentation for this class was generated from the following file:

- decimal

4.683 std::default_delete< _Tp > Struct Template Reference

Public Member Functions

- template<typename _Up , typename = typename enable_if<is_convertible<_Up∗, _Tp∗>::value>::type>
  default_delete (const default_delete< _Up > &) noexcept
- void **operator()** (_Tp ∗__ptr) const

4.683.1 Detailed Description

template<typename _Tp>struct std::default_delete< _Tp >

Primary template, default_delete.

Definition at line 54 of file unique_ptr.h.

The documentation for this struct was generated from the following file:

- unique_ptr.h

4.684 std::default_delete< _Tp[]> Struct Template Reference

Public Member Functions

- template<typename _Up , typename = typename enable_if<!__is_derived_Tp<_Up>::value>::type>
  default_delete (const default_delete< _Up[]> &) noexcept
- void **operator()** (_Tp ∗__ptr) const

- template<typename _Up >
  enable_if< __is_derived_Tp
  < _Up >::value >::type **operator()** (_Up ∗) const =delete

4.684.1 Detailed Description

template<typename _Tp>struct std::default_delete< _Tp[ ]>

Specialization, default_delete.

Definition at line 75 of file unique_ptr.h.

The documentation for this struct was generated from the following file:

- unique_ptr.h

4.685 std::defer_lock_t Struct Reference

4.685.1 Detailed Description

Do not acquire ownership of the mutex.

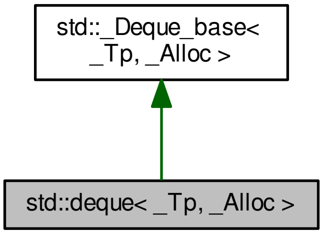
Definition at line 389 of file mutex.

The documentation for this struct was generated from the following file:

- mutex

4.686 std::deque< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::deque< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Base::const_iterator **const_iterator**

- typedef _Tp_alloc_type::const_pointer **const_pointer**
- typedef _Tp_alloc_type::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Tp_alloc_type::pointer **pointer**
- typedef _Tp_alloc_type::reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- deque ()
- deque (const allocator_type &__a)
- deque (size_type __n)
- deque (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- deque (const deque &__x)
- deque (deque &&__x)
- deque (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> deque (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- ∼deque () noexcept
- void assign (size_type __n, const value_type &__val)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- reference at (size_type __n)
- const_reference at (size_type __n) const
- reference back ()
- const_reference back () const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args> iterator emplace (iterator __position, _Args &&...__args)
- template<typename... _Args> void **emplace_back** (_Args &&...__args)
- template<typename... _Args> void **emplace_front** (_Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept

- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- iterator insert (iterator __position, const value_type &__x)
- iterator insert (iterator __position, value_type &&__x)
- void insert (iterator __p, initializer_list< value_type > __l)
- void insert (iterator __position, size_type __n, const value_type &__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void insert (iterator __position, _InputIterator __first, _InputIterator __last)
- size_type max_size () const noexcept
- deque & operator= (const deque &__x)
- deque & operator= (deque &&__x)
- deque & operator= (initializer_list< value_type > __l)
- reference operator[] (size_type __n)
- const_reference operator[] (size_type __n) const
- void pop_back ()
- void pop_front ()
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- void push_front (const value_type &__x)
- void **push_front** (value_type &&__x)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void shrink_to_fit ()
- size_type size () const noexcept
- void swap (deque &__x)

Protected Types

- enum { **_S_initial_map_size** }
- typedef _Alloc::template
  rebind< _Tp * >::other **_Map_alloc_type**
- typedef pointer * **_Map_pointer**

Protected Member Functions

- _Tp ** **_M_allocate_map** (size_t __n)
- _Tp * **_M_allocate_node** ()
- template<typename _InputIterator >
  void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _Integer >
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)

- template<typename _InputIterator >
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_create_nodes** (_Tp **__nstart, _Tp **__nfinish)
- void **_M_deallocate_map** (_Tp **__p, size_t __n)
- void **_M_deallocate_node** (_Tp *__p)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** ()
- template<typename _Alloc1 >
  void **_M_destroy_data** (iterator __first, iterator __last, const _Alloc1 &)
- void **_M_destroy_data** (iterator __first, iterator __last, const std::allocator< _Tp > &)
- void **_M_destroy_data_aux** (iterator __first, iterator __last)
- void **_M_destroy_nodes** (_Tp **__nstart, _Tp **__nfinish)
- void **_M_erase_at_begin** (iterator __pos)
- void **_M_erase_at_end** (iterator __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void _M_fill_initialize (const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- template<typename _Integer >
  void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
  void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void _M_initialize_map (size_t)
- template<typename... _Args>
  iterator **_M_insert_aux** (iterator __pos, _Args &&...__args)
- void **_M_insert_aux** (iterator __pos, size_type __n, const value_type &__x)
- template<typename _ForwardIterator >
  void **_M_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)
- template<typename _Integer >
  void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
  void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void _M_range_check (size_type __n) const
- template<typename _InputIterator >
  void **_M_range_insert_aux** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_range_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- bool **_M_shrink_to_fit** ()

- template<typename _InputIterator >
  void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)

- template<typename... _Args>
  void _M_push_back_aux (_Args &&...__args)
- template<typename... _Args>
  void _M_push_front_aux (_Args &&...__args)

- void _M_pop_back_aux ()
- void _M_pop_front_aux ()

- iterator _M_reserve_elements_at_front (size_type __n)
- iterator _M_reserve_elements_at_back (size_type __n)
- void _M_new_elements_at_front (size_type __new_elements)
- void _M_new_elements_at_back (size_type __new_elements)

- void _M_reserve_map_at_back (size_type __nodes_to_add=1)
- void _M_reserve_map_at_front (size_type __nodes_to_add=1)
- void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)

Static Protected Member Functions

- static size_t **_S_buffer_size** ()

Protected Attributes

- _Deque_impl **_M_impl**

4.686.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>class std::deque< _Tp, _Alloc >

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements`.

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a deque<Tp> manages memory. Each deque has 4 members:

- Tp** _M_map

- size_t _M_map_size

- iterator _M_start, _M_finish

map_size is at least 8. map is an array of map_size pointers-to-*nodes*. (The name map has nothing to do with the std::map class, and **nodes** should not be confused with std::list's usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator i:
  - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - i.first == ∗(i.node) (This points to the node (first Tp element).)
  - i.last == i.first + node_size
  - i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map_size).
- A pointer in the range [map, map + map_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

  Here's the magic: nothing in deque is **aware** of the discontiguous storage!

  The memory setup and layout occurs in the parent, _Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 730 of file stl_deque.h.

4.686.2 Constructor & Destructor Documentation

4.686.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> **std::deque< _Tp, _Alloc >::deque ( )** `[inline]`

Default constructor creates no elements.

Definition at line 782 of file stl_deque.h.

4.686.2.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> **std::deque< _Tp, _Alloc >::deque (const allocator_type & *__a*)** `[inline]`, `[explicit]`

Creates a deque with no elements.

Parameters

__a	An allocator object.

Definition at line 790 of file stl_deque.h.

4.686.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque (size_type __n) [inline], [explicit]

Creates a deque with default constructed elements.

Parameters

__n	The number of elements to initially create.

This constructor fills the deque with *n* default constructed elements.

Definition at line 802 of file stl_deque.h.

4.686.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque (size_type __n, const value_type & __value, const allocator_type & __a = allocator_type()) [inline]

Creates a deque with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator.

This constructor fills the deque with *__n* copies of *__value*.

Definition at line 814 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_fill_initialize().

4.686.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque (const deque< _Tp, _Alloc > & __x) [inline]

Deque copy constructor.

Parameters

__x	A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocation object used by *__x*.

Definition at line 841 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin(), and std::deque< _Tp, _Alloc >::end().

4.686.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque (deque< _Tp, _Alloc > && __x) [inline]

Deque move constructor.

Parameters

__x	A deque of identical element and allocator types.

The newly-created deque contains the exact contents of *__x*. The contents of *__x* are a valid, but unspecified deque.

Definition at line 855 of file stl_deque.h.

4.686.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::deque (initializer_list< value_type > __l, const allocator_type & __a = allocator_type()) [inline]

Builds a deque from an initializer list.

Parameters

__l	An initializer_list.
__a	An allocator object.

Create a deque consisting of copies of the elements in the initializer_list *__l*.

This will call the element type's copy constructor N times (where N is __l.size()) and do no memory reallocation.

Definition at line 869 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_initialize().

4.686.2.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> std::deque< _Tp, _Alloc >::deque (_InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type()) [inline]

Builds a deque from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__a	An allocator object.

Create a deque consisting of copies of the elements from [__first, __last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(__first,__last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 896 of file stl_deque.h.

4.686.2.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::deque< _Tp, _Alloc >::~deque ( ) [inline], [noexcept]

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 917 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin(), and std::deque< _Tp, _Alloc >::end().

4.686.3 Member Function Documentation

4.686.3.1 template<typename _Tp , typename _Alloc > void deque::_M_fill_initialize (const value_type & __value) [protected]

Fills the deque with copies of value.

Parameters

__value	Initial value.

Returns

Nothing.

Precondition

_M_start and _M_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 351 of file deque.tcc.

References std::_Destroy().

Referenced by std::deque< _Tp, _Alloc >::deque().

4.686.3.2 template<typename _Tp , typename _Alloc > void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (size_t *__num_elements*) [protected], [inherited]

Layout storage.

Parameters

__num_elements	The count of T's for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 582 of file stl_deque.h.

References std::max().

4.686.3.3 template<typename _Tp , typename _Alloc > void deque::_M_new_elements_at_back (size_type *__new_elements*) [protected]

Memory-handling helpers for the previous internal insert functions.

Definition at line 845 of file deque.tcc.

References std::size().

Referenced by std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back().

4.686.3.4 template<typename _Tp , typename _Alloc > void deque::_M_new_elements_at_front (size_type *__new_elements*) [protected]

Memory-handling helpers for the previous internal insert functions.

Definition at line 820 of file deque.tcc.

References std::size().

Referenced by std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front().

4.686.3.5 template<typename _Tp , typename _Alloc > void deque::_M_pop_back_aux ( ) [protected]

Helper functions for push_∗ and pop_∗.

Definition at line 501 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::pop_back().

4.686.3.6 template<typename _Tp , typename _Alloc > void deque::_M_pop_front_aux ( ) [protected]

Helper functions for push_∗ and pop_∗.

Definition at line 516 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::pop_front().

4.686.3.7 template<typename _Tp , typename _Alloc > template<typename... _Args> void deque::_M_push_back_aux (_Args &&... *__args*) [protected]

Helper functions for push_∗ and pop_∗.

Definition at line 435 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::push_back().

4.686.3.8 template<typename _Tp , typename _Alloc > template<typename... _Args> void deque::_M_push_front_aux (_Args &&... *__args*) [protected]

Helper functions for push_∗ and pop_∗.

Definition at line 469 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::push_front().

4.686.3.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_range_check (size_type *__n*) const [inline], [protected]

Safety check used only from at().

Definition at line 1265 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::size().

Referenced by std::deque< _Tp, _Alloc >::at().

4.686.3.10 template<typename _Tp , typename _Alloc > template<typename _InputIterator > void deque::_M_range_initialize (_InputIterator *__first,* _InputIterator *__last,* std::input_iterator_tag) [protected]

Fills the deque with whatever is in [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.

Definition at line 377 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::deque().

4.686.3.11 template<typename _Tp , typename _Alloc > template<typename _ForwardIterator > void deque::_M_range_initialize (_ForwardIterator *__first*, _ForwardIterator *__last*, std::forward_iterator_tag) [protected]

Fills the deque with whatever is in [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.

Definition at line 397 of file deque.tcc.

References std::_Destroy(), std::advance(), and std::distance().

4.686.3.12 template<typename _Tp , typename _Alloc > void deque::_M_reallocate_map (size_type *__nodes_to_add*, bool *__add_at_front*) [protected]

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 870 of file deque.tcc.

References std::max().

Referenced by std::deque< _Tp, _Alloc >::_M_reserve_map_at_back(), and std::deque< _Tp, _Alloc >::_M_reserve_map_at_front().

4.686.3.13 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back (size_type *__n*) [inline], [protected]

Memory-handling helpers for the previous internal insert functions.

Definition at line 1898 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_new_elements_at_back().

4.686.3.14 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front (size_type *__n*) [inline], [protected]

Memory-handling helpers for the previous internal insert functions.

Definition at line 1888 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_new_elements_at_front().

4.686.3.15 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (size_type *__nodes_to_add* = 1) [inline], [protected]

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 1924 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_reallocate_map().

4.686.3.16 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (size_type *__nodes_to_add* = 1) [inline], [protected]

Memory-handling helpers for the major map.

Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)

Definition at line 1932 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_reallocate_map().

4.686.3.17 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::assign (size_type *__n*, const value_type & *__val*) [inline]

Assigns a given value to a deque.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 978 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::operator=().

4.686.3.18 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void std::deque< _Tp, _Alloc >::assign (_InputIterator *__first*, _InputIterator *__last*) [inline]

Assigns a range to a deque.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a deque with copies of the elements in the range [__first,__last).

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 997 of file stl_deque.h.

4.686.3.19 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::assign (initializer_list< value_type > *__l*) [inline]

Assigns an initializer list to a deque.

Parameters

__l	An initializer_list.

This function fills a deque with copies of the elements in the initializer_list *__l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 1022 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::assign().

Referenced by std::deque< _Tp, _Alloc >::assign().

4.686.3.20 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::at (size_type *__n*) [inline]

Provides access to the data contained in the deque.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out_of_range if the check fails.

Definition at line 1284 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_check().

4.686.3.21 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::at (size_type *__n*) const [inline]

Provides access to the data contained in the deque.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out_of_range if the check fails.

Definition at line 1302 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_range_check().

4.686.3.22 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::back ( ) [inline]

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1329 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::end().

4.686.3.23 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::back ( ) const [inline]

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1341 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::end().

4.686.3.24 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1037 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::clear(), std::deque< _Tp, _Alloc >::deque(), std::deque< _Tp, _Alloc >::front(), std::deque< _Tp, _Alloc >::operator=(), std::operator==(), and std::deque< _Tp, _Alloc >::~deque().

4.686.3.25 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1045 of file stl_deque.h.

4.686.3.26 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1108 of file stl_deque.h.

4.686.3.27 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1117 of file stl_deque.h.

4.686.3.28 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::clear ( ) [inline], [noexcept]

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1616 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin().

Referenced by std::deque< _Tp, _Alloc >::operator=().

4.686.3.29 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1126 of file stl_deque.h.

4.686.3.30 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1135 of file stl_deque.h.

4.686.3.31 template<typename _Tp , typename _Alloc > template<typename... _Args> deque< _Tp, _Alloc >::iterator deque::emplace (iterator *__position*, _Args &&... *__args*)

Inserts an object in deque before specified iterator.

Parameters

__position	An iterator into the deque.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location.

Definition at line 172 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::insert().

4.686.3.32 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::deque< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the deque is empty. (Thus begin() would equal end().)

Definition at line 1228 of file stl_deque.h.

4.686.3.33 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1054 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::back(), std::deque< _Tp, _Alloc >::deque(), std::deque< _Tp, _Alloc >::operator=(), std::operator==(), and std::deque< _Tp, _Alloc >::∼deque().

4.686.3.34 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::deque< _Tp, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1063 of file stl_deque.h.

4.686.3.35 template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator *__position*)

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 194 of file deque.tcc.

References std::begin(), std::end(), and std::size().

4.686.3.36 template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::erase (iterator *__first,* iterator *__last*)

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 217 of file deque.tcc.

References std::begin(), std::end(), and std::size().

4.686.3.37 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::front ( ) [inline]

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1313 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin().

4.686.3.38 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::front ( ) const [inline]

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1321 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::begin().

4.686.3.39 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::deque< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Get a copy of the memory allocation object.

Definition at line 1028 of file stl_deque.h.

4.686.3.40 template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc >::iterator deque::insert (iterator *__position*, const value_type & *__x*)

Inserts given value into deque before specified iterator.

Parameters

__position	An iterator into the deque.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 149 of file deque.tcc.

Referenced by std::deque< _Tp, _Alloc >::resize().

4.686.3.41 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::deque< _Tp, _Alloc >::insert (iterator *__position*, value_type && *__x*) [inline]

Inserts given rvalue into deque before specified iterator.

Parameters

__position	An iterator into the deque.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1492 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::emplace().

4.686.3.42 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert (iterator *__p*, initializer_list< value_type > *__l*) [inline]

Inserts an initializer list into the deque.

Parameters

__p	An iterator into the deque.
__l	An initializer_list.

This function will insert copies of the data in the initializer_list *__l* into the deque before the location specified by *__p*. This is known as *list insert*.

Definition at line 1505 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::insert().

Referenced by std::deque< _Tp, _Alloc >::insert().

4.686.3.43 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::insert (iterator *__position,* size_type *__n,* const value_type & *__x*) [inline]

Inserts a number of copies of given data into the deque.

Parameters

__position	An iterator into the deque.
__n	Number of elements to be inserted.
__x	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *__position*.

Definition at line 1519 of file stl_deque.h.

4.686.3.44 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void std::deque< _Tp, _Alloc >::insert (iterator *__position,* _InputIterator *__first,* _InputIterator *__last*) [inline]

Inserts a range into the deque.

Parameters

__position	An iterator into the deque.
__first	An input iterator.
__last	An input iterator.

This function will insert copies of the data in the range [__first,__last) into the deque before the location specified by *__position*. This is known as *range insert*.

Definition at line 1536 of file stl_deque.h.

4.686.3.45 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::deque< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the size() of the largest possible deque.

Definition at line 1147 of file stl_deque.h.

4.686.3.46 template<typename _Tp , typename _Alloc > deque< _Tp, _Alloc > & deque::operator= (const deque< _Tp, _Alloc > & *__x*)

Deque assignment operator.

Parameters

__x	A deque of identical element and allocator types.

All the elements of *x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 93 of file deque.tcc.

References std::deque< _Tp, _Alloc >::begin(), std::deque< _Tp, _Alloc >::end(), std::deque< _Tp, _Alloc >::size(), and std::size().

4.686.3.47 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= (deque< _Tp, _Alloc > && *__x*) [inline]

Deque move assignment operator.

Parameters

__x	A deque of identical element and allocator types.

The contents of *__x* are moved into this deque (without copying). *__x* is a valid, but unspecified deque.

Definition at line 939 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::clear(), and std::deque< _Tp, _Alloc >::swap().

4.686.3.48 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> deque& std::deque< _Tp, _Alloc >::operator= (initializer_list< value_type > *__l*) [inline]

Assigns an initializer list to a deque.

Parameters

__l	An initializer_list.

This function fills a deque with copies of the elements in the initializer_list *__l*.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 960 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::assign().

4.686.3.49 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::deque< _Tp, _Alloc >::operator[] (size_type *__n*) [inline]

Subscript access to the data contained in the deque.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 1244 of file stl_deque.h.

4.686.3.50 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::deque< _Tp, _Alloc >::operator[] (size_type __n) const [inline]

Subscript access to the data contained in the deque.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 1259 of file stl_deque.h.

4.686.3.51 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::pop_back ( ) [inline]

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

Definition at line 1442 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_pop_back_aux().

4.686.3.52 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::pop_front ( ) [inline]

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

Definition at line 1421 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_pop_front_aux().

4.686.3.53 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::push_back (const value_type & __x) [inline]

Add data to the end of the deque.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1390 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_push_back_aux().

4.686.3.54 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::push_front (const value_type & __x) [inline]

Add data to the front of the deque.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1359 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::_M_push_front_aux().

4.686.3.55 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1072 of file stl_deque.h.

4.686.3.56 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1081 of file stl_deque.h.

4.686.3.57 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1090 of file stl_deque.h.

4.686.3.58 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::deque< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1099 of file stl_deque.h.

4.686.3.59 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::deque< _Tp, _Alloc >::resize (size_type __new_size) [inline]

Resizes the deque to the specified number of elements.

Parameters

__new_size	Number of elements the deque should contain.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1161 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::size().

4.686.3.60 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void **std::deque**< _Tp, _Alloc >::resize (size_type *__new_size,* const value_type & *__x*) [inline]

Resizes the deque to the specified number of elements.

Parameters

__new_size	Number of elements the deque should contain.
__x	Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1183 of file stl_deque.h.

References std::deque< _Tp, _Alloc >::insert(), and std::deque< _Tp, _Alloc >::size().

4.686.3.61 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void **std::deque**< _Tp, _Alloc >::shrink_to_fit ( ) [inline]

A non-binding request to reduce memory use.

Definition at line 1219 of file stl_deque.h.

4.686.3.62 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type **std::deque**< _Tp, _Alloc >::size ( ) const [inline], [noexcept]

Returns the number of elements in the deque.

Definition at line 1142 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::_M_range_check(), std::deque< _Tp, _Alloc >::operator=(), std::operator==(), and std::deque< _Tp, _Alloc >::resize().

4.686.3.63 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void **std::deque**< _Tp, _Alloc >::swap (**deque**< _Tp, _Alloc > & *__x*) [inline]

Swaps data with another deque.

Parameters

__x	A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(d1,d2) will feed to this function.

Definition at line 1596 of file stl_deque.h.

Referenced by std::deque< _Tp, _Alloc >::operator=(), and std::swap().

The documentation for this class was generated from the following files:

- stl_deque.h
- deque.tcc

4.687 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference

Public Types

- typedef
  _RandomNumberEngine::result_type result_type

Public Member Functions

- discard_block_engine ()
- discard_block_engine (const _RandomNumberEngine &__rng)
- discard_block_engine (_RandomNumberEngine &&__rng)
- discard_block_engine (result_type __s)
- template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>
  discard_block_engine (_Sseq &__q)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- void seed (result_type __s)
- template<typename _Sseq >
  void seed (_Sseq &__q)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr size_t **block_size**
- static constexpr size_t **used_block**

Friends

- template<typename _RandomNumberEngine1 , size_t __p1, size_t __r1, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)
- bool operator== (const discard_block_engine &__lhs, const discard_block_engine &__rhs)
- template<typename _RandomNumberEngine1 , size_t __p1, size_t __r1, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)

4.687.1 Detailed Description

template<typename _RandomNumberEngine, size_t __p, size_t __r>class std::discard_block_engine< _RandomNumberEngine, __p, __r >

Produces random numbers from some base engine by discarding blocks of data.

0 <= `__r` <= `__p`

Definition at line 854 of file random.h.

4.687.2 Member Typedef Documentation

4.687.2.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::result_type

The type of the generated random value.

Definition at line 857 of file random.h.

4.687.3 Constructor & Destructor Documentation

4.687.3.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine ( ) `[inline]`

Constructs a default discard_block_engine engine.

The underlying engine is default constructed as well.

Definition at line 872 of file random.h.

4.687.3.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (const _RandomNumberEngine & *__rng*) `[inline]`, `[explicit]`

Copy constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

Definition at line 882 of file random.h.

4.687.3.3 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (_RandomNumberEngine && *__rng*) `[inline]`, `[explicit]`

Move constructs a discard_block_engine engine.

Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

Definition at line 892 of file random.h.

4.687.3.4 template<typename _RandomNumberEngine, size_t __p, size_t __r> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (result_type __s) [inline], [explicit]

Seed constructs a discard_block_engine engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

__s	A seed value for the base class engine.

Definition at line 902 of file random.h.

4.687.3.5 template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, discard_block_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type> std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (_Sseq & __q) [inline], [explicit]

Generator construct a discard_block_engine engine.

Parameters

__q	A seed sequence.

Definition at line 915 of file random.h.

4.687.4 Member Function Documentation

4.687.4.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> const _RandomNumberEngine& std::discard_block_engine< _RandomNumberEngine, __p, __r >::base ( ) const [inline], [noexcept]

Gets a const reference to the underlying generator engine object.

Definition at line 959 of file random.h.

4.687.4.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard (unsigned long long __z) [inline]

Discard a sequence of random numbers.

Definition at line 980 of file random.h.

4.687.4.3 template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::max ( ) [inline], [static]

Gets the maximum value in the generated random number range.

Definition at line 973 of file random.h.

References std::max().

4.687.4.4 template<typename _RandomNumberEngine, size_t __p, size_t __r> static constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::min ( ) [inline], [static]

Gets the minimum value in the generated random number range.

Definition at line 966 of file random.h.

References std::min().

4.687.4.5 template<typename _RandomNumberEngine , size_t __p, size_t __r> discard_block_engine< _RandomNumberEngine, __p, __r >::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::operator() ( )

Gets the next value in the generated random number sequence.

Definition at line 688 of file bits/random.tcc.

4.687.4.6 template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed ( ) `[inline]`

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

Definition at line 924 of file random.h.

4.687.4.7 template<typename _RandomNumberEngine, size_t __p, size_t __r> void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (result_type *__s*) `[inline]`

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

Definition at line 935 of file random.h.

4.687.4.8 template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _Sseq > void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (_Sseq & *__q*) `[inline]`

Reseeds the discard_block_engine object with the given seed sequence.

Parameters

__q	A seed generator function.

Definition at line 948 of file random.h.

4.687.5 Friends And Related Function Documentation

4.687.5.1 template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1 , size_t __p1, size_t __r1, typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & *__os*, const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & *__x*) `[friend]`

Inserts the current state of a discard_block_engine random number generator engine ___`x` into the output stream ___`os`.

Parameters

__os	An output stream.
__x	A discard_block_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.687.5.2 template<typename _RandomNumberEngine, size_t __p, size_t __r> bool operator== (const discard_block_engine< _RandomNumberEngine, __p, __r > & *__lhs*, const discard_block_engine< _RandomNumberEngine, __p, __r > & *__rhs*) [friend]

Compares two discard_block_engine random number generator objects of the same type for equality.

Parameters

__lhs	A discard_block_engine random number generator object.
__rhs	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1004 of file random.h.

4.687.5.3 template<typename _RandomNumberEngine, size_t __p, size_t __r> template<typename _RandomNumberEngine1 , size_t __p1, size_t __r1, typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & *__is*, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & *__x*) [friend]

Extracts the current state of a % subtract_with_carry_engine random number generator engine `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A discard_block_engine random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.688 std::discrete_distribution< _IntType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- template<typename _InputIterator >
  discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)
- **discrete_distribution** (initializer_list< double > __wl)
- template<typename _Func >
  discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)
- **discrete_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- std::vector< double > probabilities () const
- void reset ()

Friends

- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x)
- bool operator== (const discrete_distribution &__d1, const discrete_distribution &__d2)
- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x)

4.688.1 Detailed Description

template<typename _IntType = int>class std::discrete_distribution< _IntType >

A discrete_distribution random number distribution.

The formula for the discrete probability mass function is

Definition at line 5250 of file random.h.

4.688.2 Member Typedef Documentation

4.688.2.1 template<typename _IntType = int> typedef _IntType **std::discrete_distribution**< _IntType >::**result_type**

The type of the range of the distribution.

Definition at line 5253 of file random.h.

4.688.3 Member Function Documentation

4.688.3.1 template<typename _IntType = int> **result_type std::discrete_distribution**< _IntType >::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 5370 of file random.h.

References std::vector< _Tp, _Alloc >::empty(), and std::vector< _Tp, _Alloc >::size().

4.688.3.2 template<typename _IntType = int> **result_type std::discrete_distribution**< _IntType >::min ( ) const `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 5363 of file random.h.

4.688.3.3 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > **result_type std::discrete_distribution**< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*) `[inline]`

Generating functions.

Definition at line 5381 of file random.h.

References std::discrete_distribution< _IntType >::operator()().

Referenced by std::discrete_distribution< _IntType >::operator()().

4.688.3.4 template<typename _IntType = int> **param_type std::discrete_distribution**< _IntType >::param ( ) const `[inline]`

Returns the parameter set of the distribution.

Definition at line 5348 of file random.h.

4.688.3.5 template<typename _IntType = int> void **std::discrete_distribution**< _IntType >::param (const **param_type** & *__param*) `[inline]`

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 5356 of file random.h.

4.688.3.6 template<typename _IntType = int> **std::vector**<double> **std::discrete_distribution**< _IntType >::probabilities ( ) const `[inline]`

Returns the probabilities of the distribution.

Definition at line 5338 of file random.h.

References std::vector< _Tp, _Alloc >::empty().

4.688.3.7 template<typename _IntType = int> void **std::discrete_distribution**< _IntType >::reset ( ) `[inline]`

Resets the distribution state.

Definition at line 5331 of file random.h.

4.688.4 Friends And Related Function Documentation

4.688.4.1 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > **std::basic_ostream**<_CharT, _Traits>& operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os,* const **std::discrete_distribution**< _IntType1 > & *__x*) `[friend]`

Inserts a discrete_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A discrete_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.688.4.2 template<typename _IntType = int> bool operator== (const **discrete_distribution**< _IntType > & *__d1,* const **discrete_distribution**< _IntType > & *__d2*) `[friend]`

Return true if two discrete distributions have the same parameters.

Definition at line 5416 of file random.h.

4.688.4.3 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > **std::basic_istream**<_CharT, _Traits>& operator>> (**std::basic_istream**< _CharT, _Traits > & *__is,* **std::discrete_distribution**< _IntType1 > & *__x*) `[friend]`

Extracts a discrete_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A discrete_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.689 std::discrete_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef discrete_distribution< _IntType > **distribution_type**

Public Member Functions

- template<typename _InputIterator >
  param_type (_InputIterator __wbegin, _InputIterator __wend)
- **param_type** (initializer_list< double > __wil)
- template<typename _Func >
  param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)
- **param_type** (const param_type &)=default
- param_type & **operator=** (const param_type &)=default
- std::vector< double > **probabilities** () const

Friends

- class **discrete_distribution< _IntType >**
- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.689.1 Detailed Description

template<typename _IntType = int>struct std::discrete_distribution< _IntType >::param_type

Parameter type.

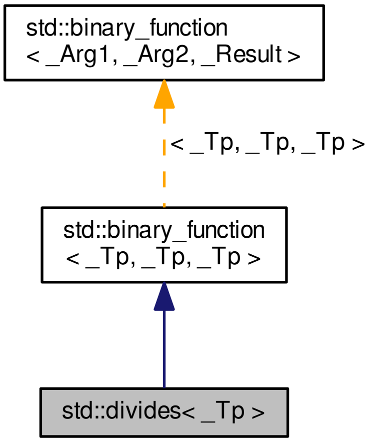
Definition at line 5259 of file random.h.

The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

4.690 std::divides< _Tp > Struct Template Reference

Inheritance diagram for std::divides< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

4.690.1 Detailed Description

template<typename _Tp>struct std::divides< _Tp >

One of the math functors.

Definition at line 167 of file stl_function.h.

4.690.2 Member Typedef Documentation

4.690.2.1 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.690.2.2 typedef _Tp **std::binary_function**< _Tp , _Tp , _Tp >**::result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.690.2.3 typedef _Tp **std::binary_function**< _Tp , _Tp , _Tp >**::second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

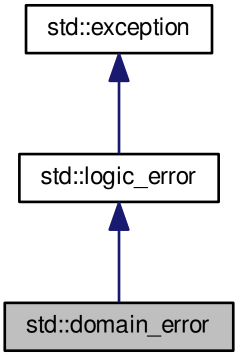
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.691 std::domain_error Class Reference

Inheritance diagram for std::domain_error:



Public Member Functions

- **domain_error** (const string &__arg)
- virtual const char * what () const noexcept

4.691.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 74 of file stdexcept.

4.691.2 Member Function Documentation

4.691.2.1 virtual const char∗ std::logic_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

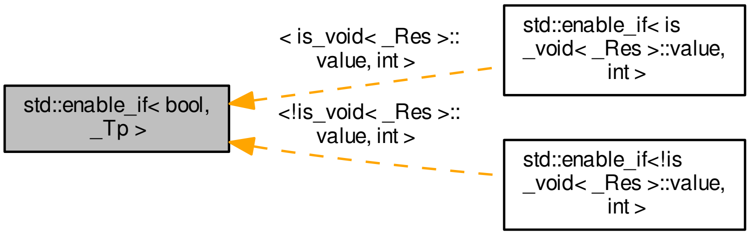
Reimplemented in std::future_error.

The documentation for this class was generated from the following file:

- stdexcept

4.692 std::enable_if< bool, _Tp > Struct Template Reference

Inheritance diagram for std::enable_if< bool, _Tp >:



4.692.1 Detailed Description

template<bool, typename _Tp = void>struct std::enable_if< bool, _Tp >

Define a member typedef `type` only if a boolean constant is true.

Definition at line 1766 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.693 std::enable_shared_from_this< _Tp > Class Template Reference

Public Member Functions

- shared_ptr< _Tp > **shared_from_this** ()
- shared_ptr< const _Tp > **shared_from_this** () const

Protected Member Functions

- **enable_shared_from_this** (const enable_shared_from_this &) noexcept
- enable_shared_from_this & **operator=** (const enable_shared_from_this &) noexcept

Friends

- template<typename _Tp1 >
  void **__enable_shared_from_this_helper** (const __shared_count<> &__pn, const enable_shared_from_this ∗-__pe, const _Tp1 ∗__px) noexcept

4.693.1 Detailed Description

template<typename _Tp>class std::enable_shared_from_this< _Tp >

Base class allowing use of member function shared_from_this.

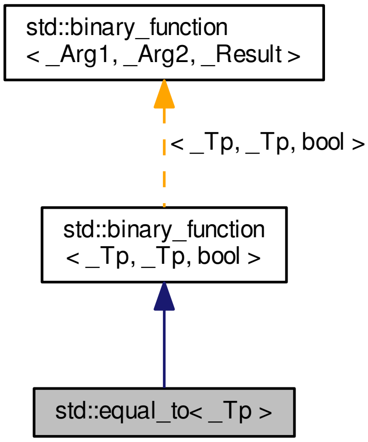
Definition at line 541 of file shared_ptr.h.

The documentation for this class was generated from the following file:

- shared_ptr.h

4.694 std::equal_to< _Tp > Struct Template Reference

Inheritance diagram for std::equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.694.1 Detailed Description

template<typename _Tp>struct std::equal_to< _Tp >

One of the comparison functors.

Definition at line 204 of file stl_function.h.

4.694.2 Member Typedef Documentation

4.694.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.694.2.2 typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.694.2.3 typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.695 std::error_category Class Reference

Public Member Functions

- **error_category** (const error_category &)=delete
- virtual error_condition **default_error_condition** (int __i) const noexcept
- virtual bool **equivalent** (int __i, const error_condition &__cond) const noexcept
- virtual bool **equivalent** (const error_code &__code, int __i) const noexcept
- virtual string **message** (int) const =0
- virtual const char ∗ **name** () const noexcept=0
- bool **operator!=** (const error_category &__other) const noexcept
- bool **operator<** (const error_category &__other) const noexcept
- error_category & **operator=** (const error_category &)=delete
- bool **operator==** (const error_category &__other) const noexcept

4.695.1 Detailed Description

error_category

Definition at line 66 of file system_error.

The documentation for this class was generated from the following file:

- system_error

4.696 std::error_code Struct Reference

Public Member Functions

- **error_code** (int __v, const error_category &__cat) noexcept
- template<typename _ErrorCodeEnum , typename = typename enable_if<is_error_code_enum<_ErrorCodeEnum>::value>::type>
  error_code (_ErrorCodeEnum __e) noexcept
- void **assign** (int __v, const error_category &__cat) noexcept
- const error_category & **category** () const noexcept
- void **clear** () noexcept
- error_condition **default_error_condition** () const noexcept
- string **message** () const
- **operator bool** () const noexcept
- template<typename _ErrorCodeEnum >
  enable_if< is_error_code_enum
  < _ErrorCodeEnum >::value,
  error_code & >::type **operator=** (_ErrorCodeEnum __e) noexcept
- int **value** () const noexcept

Friends

- class **hash< error_code >**

4.696.1 Detailed Description

error_code

Definition at line 116 of file system_error.

The documentation for this struct was generated from the following file:

- system_error

4.697 std::error_condition Struct Reference

Public Member Functions

- **error_condition** (int __v, const error_category &__cat) noexcept
- template<typename _ErrorConditionEnum , typename = typename enable_if<is_error_condition_enum<_ErrorConditionEnum>::value>::type>
  error_condition (_ErrorConditionEnum __e) noexcept
- void **assign** (int __v, const error_category &__cat) noexcept

- const error_category & **category** () const noexcept
- void **clear** () noexcept
- string **message** () const
- **operator bool** () const noexcept
- template<typename _ErrorConditionEnum >
  enable_if
  < is_error_condition_enum
  < _ErrorConditionEnum >::value,
  error_condition & >::type **operator=** (_ErrorConditionEnum __e) noexcept
- int **value** () const noexcept

4.697.1 Detailed Description

error_condition

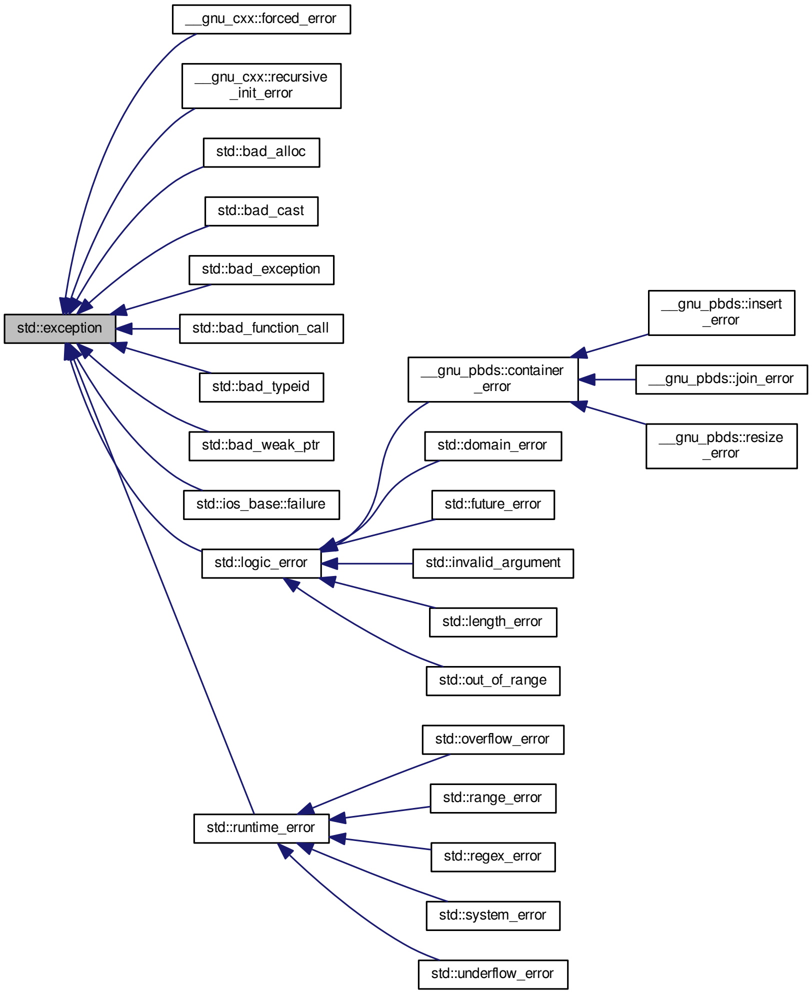
Definition at line 193 of file system_error.

The documentation for this struct was generated from the following file:

- system_error

4.698 std::exception Class Reference

Inheritance diagram for std::exception:



Public Member Functions

- virtual const char ∗ what () const noexcept

4.698.1 Detailed Description

Base class for all library exceptions.

This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 60 of file exception.

4.698.2 Member Function Documentation

4.698.2.1 virtual const char∗ std::exception::what ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error.

Reimplemented in std::bad_function_call, std::ios_base::failure, std::bad_typeid, std::bad_cast, std::runtime_error, std::future_error, std::bad_exception, std::logic_error, std::bad_weak_ptr, and std::bad_alloc.

The documentation for this class was generated from the following file:

- exception

4.699 std::exponential_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- exponential_distribution (const result_type &__lambda=result_type(1))
- **exponential_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType lambda () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const exponential_distribution &__d1, const exponential_distribution &__d2)

4.699.1 Detailed Description

template<typename _RealType = double>class std::exponential_distribution< _RealType >

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is $p(x|\lambda) = \lambda e^{-\lambda x}$.

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	$zero$
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Table 1849: Distribution Statistics

Definition at line 4645 of file random.h.

4.699.2 Member Typedef Documentation

4.699.2.1 template<typename _RealType = double> typedef _RealType std::exponential_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 4648 of file random.h.

4.699.3 Constructor & Destructor Documentation

4.699.3.1 template<typename _RealType = double> std::exponential_distribution< _RealType >::exponential_distribution (const result_type & __*lambda* = result_type `(1)`) `[inline]`, `[explicit]`

Constructs an exponential distribution with inverse scale parameter λ.

Definition at line 4683 of file random.h.

4.699.4 Member Function Documentation

4.699.4.1 template<typename _RealType = double> _RealType std::exponential_distribution< _RealType >::lambda ( ) const `[inline]`

Returns the inverse scale parameter of the distribution.

Definition at line 4704 of file random.h.

4.699.4.2 template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4733 of file random.h.

References std::numeric_limits< _Tp >::max().

4.699.4.3 template<typename _RealType = double> result_type std::exponential_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4726 of file random.h.

4.699.4.4 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::exponential_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 4741 of file random.h.

References std::exponential_distribution< _RealType >::operator()().

Referenced by std::exponential_distribution< _RealType >::operator()().

4.699.4.5 template<typename _RealType = double> param_type std::exponential_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 4711 of file random.h.

Referenced by std::operator>>().

4.699.4.6 template<typename _RealType = double> void std::exponential_distribution< _RealType >::param (const param_type & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 4719 of file random.h.

4.699.4.7 template<typename _RealType = double> void std::exponential_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4698 of file random.h.

4.699.5 Friends And Related Function Documentation

4.699.5.1 template<typename _RealType = double> bool operator== (const exponential_distribution< _RealType > & *__d1*, const exponential_distribution< _RealType > & *__d2*) [friend]

Return true if two exponential distributions have the same parameters.

Definition at line 4781 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.700 std::exponential_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef
  exponential_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __lambda=_RealType(1))
- _RealType **lambda** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.700.1 Detailed Description

template<typename _RealType = double>struct std::exponential_distribution< _RealType >::param_type

Parameter type.

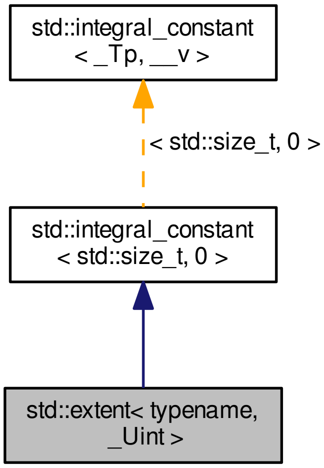
Definition at line 4654 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.701 std::extent< typename, _Uint > Struct Template Reference

Inheritance diagram for std::extent< typename, _Uint >:



Public Types

- typedef integral_constant
  < std::size_t, __v > **type**
- typedef std::size_t **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr std::size_t **value**

4.701.1 Detailed Description

template<typename, unsigned _Uint>struct std::extent< typename, _Uint >

extent

Definition at line 1256 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.702 std::extreme_value_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **extreme_value_distribution** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **extreme_value_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType a () const
- _RealType b () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const extreme_value_distribution &__d1, const extreme_value_distribution &__d2)

4.702.1 Detailed Description

template<typename _RealType = double>class std::extreme_value_distribution< _RealType >

A extreme_value_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b}\exp(\frac{a-x}{b} - \exp(\frac{a-x}{b}))$$

Definition at line 5050 of file random.h.

4.702.2 Member Typedef Documentation

4.702.2.1 template<typename _RealType = double> typedef _RealType std::extreme_value_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 5053 of file random.h.

4.702.3 Member Function Documentation

4.702.3.1 template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::a ( ) const [inline]

Return the a parameter of the distribution.

Definition at line 5108 of file random.h.

4.702.3.2 template<typename _RealType = double> _RealType std::extreme_value_distribution< _RealType >::b ( ) const [inline]

Return the b parameter of the distribution.

Definition at line 5115 of file random.h.

4.702.3.3 template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 5144 of file random.h.

References std::numeric_limits< _Tp >::max().

4.702.3.4 template<typename _RealType = double> result_type std::extreme_value_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 5137 of file random.h.

References std::numeric_limits< _Tp >::min().

4.702.3.5 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::extreme_value_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 5152 of file random.h.

References std::extreme_value_distribution< _RealType >::operator()().

Referenced by std::extreme_value_distribution< _RealType >::operator()().

4.702.3.6 template<typename _RealType = double> param_type std::extreme_value_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 5122 of file random.h.

Referenced by std::operator>>().

4.702.3.7 template<typename _RealType = double> void **std::extreme_value_distribution**< _RealType >::param (const **param_type** & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 5130 of file random.h.

4.702.3.8 template<typename _RealType = double> void **std::extreme_value_distribution**< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 5101 of file random.h.

4.702.4 Friends And Related Function Documentation

4.702.4.1 template<typename _RealType = double> bool operator== (const **extreme_value_distribution**< _RealType > & *__d1,* const **extreme_value_distribution**< _RealType > & *__d2*) [friend]

Return true if two extreme value distributions have the same parameters.

Definition at line 5187 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.703 std::extreme_value_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef extreme_value_distribution < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.703.1 Detailed Description

template<typename _RealType = double>struct std::extreme_value_distribution< _RealType >::param_type

Parameter type.

Definition at line 5059 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.704 std::fisher_f_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **fisher_f_distribution** (_RealType __m=_RealType(1), _RealType __n=_RealType(1))
- **fisher_f_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **m** () const
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::fisher_f_distribution< _RealType1 > &__x)
- bool operator== (const fisher_f_distribution &__d1, const fisher_f_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::fisher_f_distribution< _RealType1 > &__x)

4.704.1 Detailed Description

template<typename _RealType = double>class std::fisher_f_distribution< _RealType >

A fisher_f_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m,n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)}(\frac{m}{n})^{m/2}x^{(m/2)-1}(1+\frac{mx}{n})^{-(m+n)/2}$$

Definition at line 3130 of file random.h.

4.704.2 Member Typedef Documentation

4.704.2.1 template<typename _RealType = double> typedef _RealType std::fisher_f_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 3133 of file random.h.

4.704.3 Member Function Documentation

4.704.3.1 template<typename _RealType = double> result_type std::fisher_f_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3224 of file random.h.

References std::numeric_limits< _Tp >::max().

4.704.3.2 template<typename _RealType = double> result_type std::fisher_f_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3217 of file random.h.

4.704.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::fisher_f_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 3232 of file random.h.

4.704.3.4 template<typename _RealType = double> param_type std::fisher_f_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 3202 of file random.h.

4.704.3.5 template<typename _RealType = double> void std::fisher_f_distribution< _RealType >::param (const param_type & __param) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 3210 of file random.h.

4.704.3.6 template<typename _RealType = double> void std::fisher_f_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 3181 of file random.h.

References std::gamma_distribution< _RealType >::reset().

4.704.4 Friends And Related Function Documentation

4.704.4.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::fisher_f_distribution< _RealType1 > & __x) [friend]

Inserts a fisher_f_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A fisher_f_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.704.4.2 template<typename _RealType = double> bool operator== (const fisher_f_distribution< _RealType > & __d1, const fisher_f_distribution< _RealType > & __d2) [friend]

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3280 of file random.h.

4.704.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & __is, std::fisher_f_distribution< _RealType1 > & __x) [friend]

Extracts a fisher_f_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A fisher_f_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.705 std::fisher_f_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef fisher_f_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(1), _RealType __n=_RealType(1))
- _RealType **m** () const
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.705.1 Detailed Description

template<typename _RealType = double>struct std::fisher_f_distribution< _RealType >::param_type

Parameter type.

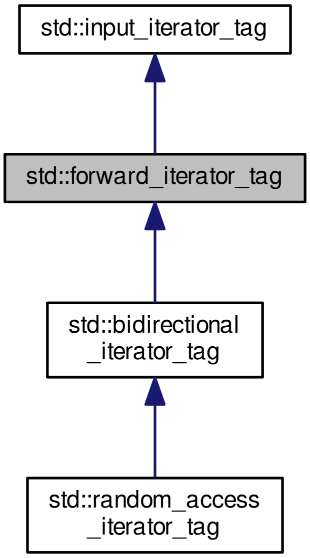
Definition at line 3139 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.706 std::forward_iterator_tag Struct Reference

Inheritance diagram for std::forward_iterator_tag:



4.706.1 Detailed Description

Forward iterators support a superset of input iterator operations.

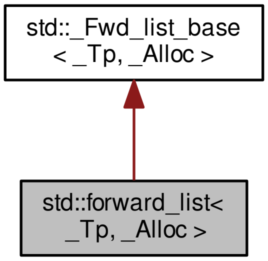
Definition at line 95 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.707 std::forward_list< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef
  _Fwd_list_const_iterator< _Tp > **const_iterator**
- typedef
  _Alloc_traits::const_pointer **const_pointer**
- typedef
  _Alloc_traits::const_reference **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Fwd_list_iterator< _Tp > **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- forward_list (const _Alloc &__al=_Alloc())
- forward_list (const forward_list &__list, const _Alloc &__al)
- forward_list (forward_list &&__list, const _Alloc &__al) noexcept(_Node_alloc_traits
- forward_list (size_type __n, const _Alloc &__al=_Alloc())
- forward_list (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- forward_list (const forward_list &__list)
- forward_list (forward_list &&__list) noexcept
- forward_list (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- ∼forward_list () noexcept
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)

- void assign (size_type __n, const _Tp &__val)
- void assign (std::initializer_list< _Tp > __il)
- iterator before_begin () noexcept
- const_iterator before_begin () const noexcept
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const_iterator cbefore_begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  iterator emplace_after (const_iterator __pos, _Args &&...__args)
- template<typename... _Args>
  void emplace_front (_Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- iterator erase_after (const_iterator __pos)
- iterator erase_after (const_iterator __pos, const_iterator __last)
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- iterator insert_after (const_iterator __pos, const _Tp &__val)
- iterator **insert_after** (const_iterator __pos, _Tp &&__val)
- iterator insert_after (const_iterator __pos, size_type __n, const _Tp &__val)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- iterator insert_after (const_iterator __pos, std::initializer_list< _Tp > __il)
- size_type max_size () const noexcept
- void merge (forward_list &&__list)
- void **merge** (forward_list &__list)
- template<typename _Comp >
  void merge (forward_list &&__list, _Comp __comp)
- template<typename _Comp >
  void **merge** (forward_list &__list, _Comp __comp)
- forward_list & operator= (const forward_list &__list)
- forward_list & operator= (forward_list &&__list) noexcept(_Node_alloc_traits
- forward_list & operator= (std::initializer_list< _Tp > __il)
- void pop_front ()
- void push_front (const _Tp &__val)
- void **push_front** (_Tp &&__val)
- void remove (const _Tp &__val)
- template<typename _Pred >
  void remove_if (_Pred __pred)
- void resize (size_type __sz)
- void resize (size_type __sz, const value_type &__val)
- void reverse () noexcept
- void sort ()
- template<typename _Comp >
  void sort (_Comp __comp)
- void splice_after (const_iterator __pos, forward_list &&__list)

- void **splice_after** (const_iterator __pos, forward_list &__list)
- void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __i)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __i)
- void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last)
- void **splice_after** (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last)
- void swap (forward_list &__list) noexcept(_Node_alloc_traits
- void unique ()
- template<typename _BinPred >
  void unique (_BinPred __binary_pred)

Private Member Functions

- template<typename... _Args>
  _Node ∗ **_M_create_node** (_Args &&...__args)
- _Fwd_list_node_base ∗ **_M_erase_after** (_Fwd_list_node_base ∗__pos)
- _Fwd_list_node_base ∗ **_M_erase_after** (_Fwd_list_node_base ∗__pos, _Fwd_list_node_base ∗__last)
- _Node ∗ **_M_get_node** ()
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- template<typename... _Args>
  _Fwd_list_node_base ∗ **_M_insert_after** (const_iterator __pos, _Args &&...__args)
- void **_M_put_node** (_Node ∗__p)

Private Attributes

- _Fwd_list_impl **_M_impl**

4.707.1 Detailed Description

template<typename _Tp, typename _Alloc = allocator<_Tp>>class std::forward_list< _Tp, _Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator`[].

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike std::vector and std::deque, random-access iterators are not provided, so subscripting ([]) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, std::forward_list provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

Definition at line 408 of file forward_list.h.

4.707.2 Constructor & Destructor Documentation

4.707.2.1 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (const _Alloc & *__al* = `_Alloc()`) `[inline]`, `[explicit]`

Creates a forward_list with no elements.

Parameters

__al	An allocator object.

Definition at line 440 of file forward_list.h.

4.707.2.2 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (const **forward_list**< _Tp, _Alloc > & *__list*, const _Alloc & *__al*) `[inline]`

Copy constructor with allocator argument.

Parameters

__list	Input list to copy.
__al	An allocator object.

Definition at line 449 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::begin(), and std::forward_list< _Tp, _Alloc >::end().

4.707.2.3 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (**forward_list**< _Tp, _Alloc > && *__list*, const _Alloc & *__al*) `[inline]`, `[noexcept]`

Move constructor with allocator argument.

Parameters

__list	Input list to move.
__al	An allocator object.

Definition at line 458 of file forward_list.h.

4.707.2.4 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (size_type *__n*, const _Alloc & *__al* = `_Alloc()`) `[inline]`, `[explicit]`

Creates a forward_list with default constructed elements.

Parameters

__n	The number of elements to initially create.

This constructor creates the forward_list with *__n* default constructed elements.

Definition at line 471 of file forward_list.h.

4.707.2.5 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (size_type *__n*, const _Tp & *__value*, const _Alloc & *__al* = `_Alloc()`) `[inline]`

Creates a forward_list with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__al	An allocator object.

This constructor fills the forward_list with __*n* copies of __*value*.

Definition at line 484 of file forward_list.h.

4.707.2.6 template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (_InputIterator *__first*, _InputIterator *__last*, const _Alloc & *__al* = `_Alloc()`) `[inline]`

Builds a forward_list from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__al	An allocator object.

Create a forward_list consisting of copies of the elements from [__*first*,__*last*). This is linear in N (where N is distance(__*first*,__*last*)).

Definition at line 501 of file forward_list.h.

4.707.2.7 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (const **forward_list**< _Tp, _Alloc > & *__list*) `[inline]`

The forward_list copy constructor.

Parameters

__list	A forward_list of identical element and allocator types.

Definition at line 511 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::begin(), and std::forward_list< _Tp, _Alloc >::end().

4.707.2.8 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (**forward_list**< _Tp, _Alloc > && *__list*) `[inline]`, `[noexcept]`

The forward_list move constructor.

Parameters

__list	A forward_list of identical element and allocator types.

The newly-created forward_list contains the exact contents of __*list*. The contents of __*list* are a valid, but unspecified forward_list.

Definition at line 525 of file forward_list.h.

4.707.2.9 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::**forward_list** (**std::initializer_list**< _Tp > *__il*, const _Alloc & *__al* = `_Alloc()`) `[inline]`

Builds a forward_list from an initializer_list.

Parameters

__il	An initializer_list of value_type.
__al	An allocator object.

Create a forward_list consisting of copies of the elements in the initializer_list *__il*. This is linear in __il.size().

Definition at line 536 of file forward_list.h.

4.707.2.10 template<typename _Tp, typename _Alloc = allocator<_Tp>> **std::forward_list**< _Tp, _Alloc >::∼**forward_list** ( ) [inline], [noexcept]

The forward_list dtor.

Definition at line 544 of file forward_list.h.

4.707.3 Member Function Documentation

4.707.3.1 template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void **std::forward_list**< _Tp, _Alloc >::assign (_InputIterator *__first*, _InputIterator *__last*) [inline]

Assigns a range to a forward_list.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a forward_list with copies of the elements in the range [*__first*,*__last*).

Note that the assignment completely changes the forward_list and that the number of elements of the resulting forward_list is the same as the number of elements assigned. Old data is lost.

Definition at line 609 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=().

4.707.3.2 template<typename _Tp, typename _Alloc = allocator<_Tp>> void **std::forward_list**< _Tp, _Alloc >::assign (size_type *__n*, const _Tp & *__val*) [inline]

Assigns a given value to a forward_list.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a forward_list with *__n* copies of the given value. Note that the assignment completely changes the forward_list, and that the resulting forward_list has __n elements. Old data is lost.

Definition at line 626 of file forward_list.h.

4.707.3.3 template<typename _Tp, typename _Alloc = allocator<_Tp>> void **std::forward_list**< _Tp, _Alloc >::assign (**std::initializer_list**< _Tp > *__il*) [inline]

Assigns an initializer_list to a forward_list.

Parameters

__il	An initializer_list of value_type.

Replace the contents of the forward_list with copies of the elements in the initializer_list *__il*. This is linear in il.size().

Definition at line 638 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::assign().

Referenced by std::forward_list< _Tp, _Alloc >::assign().

4.707.3.4 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) `[inline]`, `[noexcept]`

Returns a read/write iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 653 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::insert_after().

4.707.3.5 template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::before_begin ( ) const `[inline]`, `[noexcept]`

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 662 of file forward_list.h.

4.707.3.6 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::begin ( ) `[inline]`, `[noexcept]`

Returns a read/write iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 670 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::forward_list().

4.707.3.7 template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::begin ( ) const `[inline]`, `[noexcept]`

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 679 of file forward_list.h.

4.707.3.8 template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin ( ) const `[inline]`, `[noexcept]`

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 715 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::emplace_front(), and std::forward_list< _Tp, _Alloc >::push_front().

4.707.3.9 template<typename _Tp, typename _Alloc = allocator<_Tp>> **const_iterator std::forward_list**< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Definition at line 706 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::operator==().

4.707.3.10 template<typename _Tp, typename _Alloc = allocator<_Tp>> **const_iterator std::forward_list**< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 724 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::operator=(), and std::operator==().

4.707.3.11 template<typename _Tp, typename _Alloc = allocator<_Tp>> void **std::forward_list**< _Tp, _Alloc >::clear ( ) [inline], [noexcept]

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1027 of file forward_list.h.

4.707.3.12 template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> **iterator std::forward_list**< _Tp, _Alloc >::emplace_after (**const_iterator** *__pos*, _Args &&... *__args*) [inline]

Constructs object in forward_list after the specified iterator.

Parameters

__pos	A const_iterator into the forward_list.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 837 of file forward_list.h.

4.707.3.13 template<typename _Tp, typename _Alloc = allocator<_Tp>> template<typename... _Args> void **std::forward_list**< _Tp, _Alloc >::emplace_front (_Args &&... *__args*) [inline]

Constructs object in forward_list at the front of the list.

Parameters

__args	Arguments.

This function will insert an object of type Tp constructed with Tp(std::forward<Args>(args)...) at the front of the list Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 781 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::cbefore_begin().

4.707.3.14 template<typename _Tp, typename _Alloc = allocator<_Tp>> bool std::forward_list< _Tp, _Alloc >::empty ( ) const `[inline]`, `[noexcept]`

Returns true if the forward_list is empty. (Thus begin() would equal end().)

Definition at line 732 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::insert_after().

4.707.3.15 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::end ( ) `[inline]`, `[noexcept]`

Returns a read/write iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 688 of file forward_list.h.

Referenced by std::forward_list< _Tp, _Alloc >::forward_list(), and std::forward_list< _Tp, _Alloc >::insert_after().

4.707.3.16 template<typename _Tp, typename _Alloc = allocator<_Tp>> const_iterator std::forward_list< _Tp, _Alloc >::end ( ) const `[inline]`, `[noexcept]`

Returns a read-only iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Definition at line 697 of file forward_list.h.

4.707.3.17 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::erase_after (const_iterator *__pos*) `[inline]`

Removes the element pointed to by the iterator following `pos`.

Parameters

__pos	Iterator pointing before element to be erased.

Returns

An iterator pointing to the element following the one that was erased, or end() if no such element exists.

This function will erase the element at the given position and thus shorten the forward_list by one.

Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 940 of file forward_list.h.

4.707.3.18 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::erase_after (const_iterator *__pos*, const_iterator *__last*) `[inline]`

Remove a range of elements.

Parameters

__pos	Iterator pointing before the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

@ __last.

This function will erase the elements in the range (__pos,__last) and shorten the forward_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 963 of file forward_list.h.

4.707.3.19 template<typename _Tp, typename _Alloc = allocator<_Tp>> reference std::forward_list< _Tp, _Alloc >::front ( ) [inline]

Returns a read/write reference to the data at the first element of the forward_list.

Definition at line 749 of file forward_list.h.

4.707.3.20 template<typename _Tp, typename _Alloc = allocator<_Tp>> const_reference std::forward_list< _Tp, _Alloc >::front ( ) const [inline]

Returns a read-only (constant) reference to the data at the first element of the forward_list.

Definition at line 760 of file forward_list.h.

4.707.3.21 template<typename _Tp, typename _Alloc = allocator<_Tp>> allocator_type std::forward_list< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Get a copy of the memory allocation object.

Definition at line 643 of file forward_list.h.

4.707.3.22 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::insert_after (const_iterator *__pos*, const _Tp & *__val*) [inline]

Inserts given value into forward_list after specified iterator.

Parameters

__pos	An iterator into the forward_list.
__val	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 854 of file forward_list.h.

4.707.3.23 template<typename _Tp, typename _Alloc > forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (const_iterator *__pos,* size_type *__n,* const _Tp & *__val*)

Inserts a number of copies of given data into the forward_list.

Parameters

__pos	An iterator into the forward_list.
__n	Number of elements to be inserted.
__val	Data to be inserted.

Returns

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 272 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::before_begin(), and std::forward_list< _Tp, _Alloc >::end().

4.707.3.24 template<typename _Tp , typename _Alloc > template<typename _InputIterator , typename > forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (const_iterator *__pos,* _InputIterator *__first,* _InputIterator *__last*)

Inserts a range into the forward_list.

Parameters

__pos	An iterator into the forward_list.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator pointing to the last inserted element or *__pos* if *__first* == *__last*.

This function will insert copies of the data in the range [*__first*,*__last*) into the forward_list after the location specified by *__pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 287 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::before_begin(), std::forward_list< _Tp, _Alloc >::empty(), and std::forward_list< _Tp, _Alloc >::end().

4.707.3.25 template<typename _Tp, typename _Alloc = allocator<_Tp>> iterator std::forward_list< _Tp, _Alloc >::insert_after (const_iterator *__pos*, std::initializer_list< _Tp > *__il*) [inline]

Inserts the contents of an initializer_list into forward_list after the specified iterator.

Parameters

__pos	An iterator into the forward_list.
__il	An initializer_list of value_type.

Returns

An iterator pointing to the last inserted element or *__pos* if *__il* is empty.

This function will insert copies of the data in the initializer_list *__il* into the forward_list before the location specified by *__pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 919 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::insert_after().

Referenced by std::forward_list< _Tp, _Alloc >::insert_after().

4.707.3.26 template<typename _Tp, typename _Alloc = allocator<_Tp>> size_type std::forward_list< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the largest possible number of elements of forward_list.

Definition at line 739 of file forward_list.h.

References std::allocator_traits< _Alloc >::max_size().

4.707.3.27 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::merge (forward_list< _Tp, _Alloc > && *__list*) [inline]

Merge sorted lists.

Parameters

__list	Sorted list to merge.

Assumes that both *list* and this list are sorted according to operator<(). Merges elements of *__list* into this list in sorted order, leaving *__list* empty when complete. Elements in this list precede elements in *__list* that are equal.

Definition at line 1165 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::merge().

Referenced by std::forward_list< _Tp, _Alloc >::merge().

4.707.3.28 template<typename _Tp , typename _Alloc > template<typename _Comp > void forward_list::merge (forward_list< _Tp, _Alloc > && *__list*, _Comp *__comp*)

Merge sorted lists according to comparison function.

Parameters

__list	Sorted list to merge.
__comp	Comparison function defining sort order.

Assumes that both *__list* and this list are sorted according to comp. Merges elements of *__list* into this list in sorted order, leaving *__list* empty when complete. Elements in this list precede elements in *__list* that are equivalent according to comp().

Definition at line 365 of file forward_list.tcc.

4.707.3.29 template< typename _Tp , typename _Alloc > forward_list< _Tp, _Alloc > & forward_list::operator= (const forward_list< _Tp, _Alloc > & *__list*)

The forward_list assignment operator.

Parameters

__list	A forward_list of identical element and allocator types.

All the elements of *__list* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 156 of file forward_list.tcc.

References std::forward_list< _Tp, _Alloc >::cbegin(), and std::forward_list< _Tp, _Alloc >::cend().

4.707.3.30 template< typename _Tp, typename _Alloc = allocator< _Tp>> forward_list& std::forward_list< _Tp, _Alloc >::operator= (forward_list< _Tp, _Alloc > && *__list*) `[inline]`, `[noexcept]`

The forward_list move assignment operator.

Parameters

__list	A forward_list of identical element and allocator types.

The contents of *__list* are moved into this forward_list (without copying, if the allocators permit it). *__list* is a valid, but unspecified forward_list

Definition at line 568 of file forward_list.h.

4.707.3.31 template< typename _Tp, typename _Alloc = allocator< _Tp>> forward_list& std::forward_list< _Tp, _Alloc >::operator= (std::initializer_list< _Tp > *__il*) `[inline]`

The forward_list initializer list assignment operator.

Parameters

__il	An initializer_list of value_type.

Replace the contents of the forward_list with copies of the elements in the initializer_list *__il*. This is linear in __il.size().

Definition at line 588 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::assign().

4.707.3.32 template< typename _Tp, typename _Alloc = allocator< _Tp>> void std::forward_list< _Tp, _Alloc >::pop_front ( ) `[inline]`

Removes first element.

This is a typical stack operation. It shrinks the forward_list by one. Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

Definition at line 819 of file forward_list.h.

4.707.3.33 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::push_front (const _Tp & __val) [inline]

Add data to the front of the forward_list.

Parameters

__val	Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward_list and assigns the given data to it. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 796 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::cbefore_begin().

4.707.3.34 template<typename _Tp, typename _Alloc > void forward_list::remove (const _Tp & __val)

Remove all elements equal to value.

Parameters

__val	The value to remove.

Removes every element in the list equal to *__val*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 300 of file forward_list.tcc.

References std::__addressof().

4.707.3.35 template<typename _Tp , typename _Alloc > template<typename _Pred > void forward_list::remove_if (_Pred __pred)

Remove all elements satisfying a predicate.

Parameters

__pred	Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 328 of file forward_list.tcc.

4.707.3.36 template<typename _Tp , typename _Alloc > void forward_list::resize (size_type __sz)

Resizes the forward_list to the specified number of elements.

Parameters

__sz	Number of elements the forward_list should contain.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current number of elements the forward_list is truncated, otherwise the forward_list is extended and the new elements are default constructed.

Definition at line 198 of file forward_list.tcc.

References std::end().

4.707.3.37 template<typename _Tp , typename _Alloc > void forward_list::resize (size_type *__sz*, const value_type & *__val*)

Resizes the forward_list to the specified number of elements.

Parameters

__sz	Number of elements the forward_list should contain.
__val	Data with which new elements should be populated.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current number of elements the forward_list is truncated, otherwise the forward_list is extended and new elements are populated with given data.

Definition at line 217 of file forward_list.tcc.

References std::end().

4.707.3.38 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::reverse ( ) [inline], [noexcept]

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1218 of file forward_list.h.

4.707.3.39 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::sort ( ) [inline]

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1199 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::sort().

Referenced by std::forward_list< _Tp, _Alloc >::sort().

4.707.3.40 template<typename _Tp , class _Alloc > template<typename _Comp > void forward_list::sort (_Comp *__comp*)

Sort the forward_list using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 411 of file forward_list.tcc.

4.707.3.41 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (const_iterator *__pos*, forward_list< _Tp, _Alloc > && *__list*) [inline]

Insert contents of another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.

The elements of *list* are inserted in constant time after the element referenced by *pos*. *list* becomes an empty list.

Requires this != *x*.

Definition at line 1044 of file forward_list.h.

4.707.3.42 template<typename _Tp , typename _Alloc > void forward_list::splice_after (const_iterator *__pos*, forward_list< _Tp, _Alloc > && *__list*, const_iterator *__i*)

Insert element from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__i	Iterator referencing the element before the element to move.

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 255 of file forward_list.tcc.

4.707.3.43 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::splice_after (const_iterator *__pos*, forward_list< _Tp, _Alloc > && , const_iterator *__before*, const_iterator *__last*) [inline]

Insert range from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__before	Iterator referencing before the start of range in list.
__last	Iterator referencing the end of range in list.

Removes elements in the range (__before,__last) and inserts them after *__pos* in constant time.

Undefined if *__pos* is in (__before,__last).

Definition at line 1087 of file forward_list.h.

4.707.3.44 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::swap (forward_list< _Tp, _Alloc > & *__list*) [inline], [noexcept]

Swaps data with another forward_list.

Parameters

__list	A forward_list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global std::swap() function is specialized such that std::swap(l1,l2) will feed to this function.

Definition at line 980 of file forward_list.h.

Referenced by std::swap().

4.707.3.45 template<typename _Tp, typename _Alloc = allocator<_Tp>> void std::forward_list< _Tp, _Alloc >::unique ( ) [inline]

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1136 of file forward_list.h.

References std::forward_list< _Tp, _Alloc >::unique().

Referenced by std::forward_list< _Tp, _Alloc >::unique().

4.707.3.46 template<typename _Tp , typename _Alloc > template<typename _BinPred > void forward_list::unique (_BinPred *__binary_pred*)

Remove consecutive elements satisfying a predicate.

Parameters

__binary_pred	Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 344 of file forward_list.tcc.

References std::begin(), and std::end().

The documentation for this class was generated from the following files:

- forward_list.h
- forward_list.tcc

4.708 std::fpos< _StateT > Class Template Reference

Public Member Functions

- fpos (streamoff __off)
- operator streamoff () const
- fpos operator+ (streamoff __off) const
- fpos & operator+= (streamoff __off)
- fpos operator- (streamoff __off) const
- streamoff operator- (const fpos &__other) const
- fpos & operator-= (streamoff __off)
- void state (_StateT __st)
- _StateT state () const

4.708.1 Detailed Description

template<typename _StateT>class std::fpos< _StateT >

Class representing stream positions.

The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

Parameters

StateT	Type passed to and returned from state().

Definition at line 112 of file postypes.h.

4.708.2 Constructor & Destructor Documentation

4.708.2.1 template<typename _StateT> std::fpos< _StateT >::fpos (streamoff *__off*) [inline]

Construct position from offset.

Definition at line 133 of file postypes.h.

4.708.3 Member Function Documentation

4.708.3.1 template<typename _StateT> std::fpos< _StateT >::operator streamoff ( ) const [inline]

Convert to streamoff.

Definition at line 137 of file postypes.h.

4.708.3.2 template<typename _StateT> fpos std::fpos< _StateT >::operator+ (streamoff *__off*) const [inline]

Add position and offset.

Definition at line 178 of file postypes.h.

4.708.3.3 template<typename _StateT> fpos& std::fpos< _StateT >::operator+= (streamoff *__off*) [inline]

Add offset to this position.

Definition at line 154 of file postypes.h.

4.708.3.4 template<typename _StateT> fpos std::fpos< _StateT >::operator- (streamoff *__off*) const [inline]

Subtract offset from position.

Definition at line 192 of file postypes.h.

4.708.3.5 template<typename _StateT> streamoff std::fpos< _StateT >::operator- (const fpos< _StateT > & *__other*) const [inline]

Subtract position to return offset.

Definition at line 205 of file postypes.h.

4.708.3.6 template<typename _StateT> fpos& std::fpos< _StateT >::operator-= (streamoff *__off*) [inline]

Subtract offset from this position.

Definition at line 165 of file postypes.h.

4.708.3.7 template<typename _StateT> void std::fpos< _StateT >::state (_StateT *__st*) [inline]

Remember the value of *st*.

Definition at line 141 of file postypes.h.

4.708.3.8 template<typename _StateT> _StateT **std::fpos**< _StateT >::state ( ) const [inline]

Return the last set value of *st*.

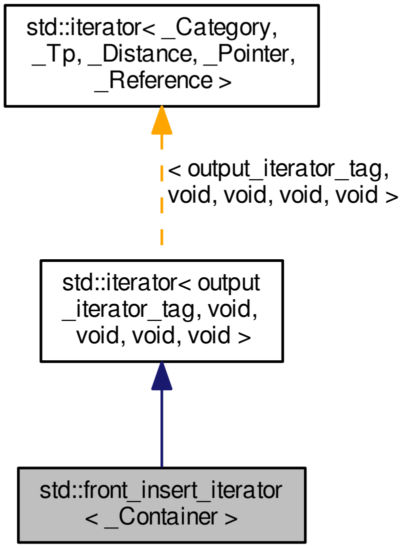
Definition at line 146 of file postypes.h.

The documentation for this class was generated from the following file:

- postypes.h

4.709 std::front_insert_iterator< _Container > Class Template Reference

Inheritance diagram for std::front_insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- front_insert_iterator (_Container &__x)
- front_insert_iterator & operator* ()

- front_insert_iterator & operator++ ()
- front_insert_iterator operator++ (int)
- front_insert_iterator & operator= (const typename _Container::value_type &__value)
- front_insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container ∗ **container**

4.709.1 Detailed Description

template<typename _Container>class std::front_insert_iterator< _Container >

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using push_front.

Tip: Using the front_inserter function to create these iterators can save typing.

Definition at line 493 of file stl_iterator.h.

4.709.2 Member Typedef Documentation

4.709.2.1 template<typename _Container > typedef _Container std::front_insert_iterator< _Container >::container_type

A nested typedef for the type of whatever container you used.

Definition at line 501 of file stl_iterator.h.

4.709.2.2 typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.709.2.3 typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.709.2.4 typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.709.2.5 typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.709.2.6 typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.709.3 Constructor & Destructor Documentation

4.709.3.1 template<typename _Container > std::front_insert_iterator< _Container >::front_insert_iterator (_Container & __x) [inline], [explicit]

The only way to create this iterator is with a container.

Definition at line 504 of file stl_iterator.h.

4.709.4 Member Function Documentation

4.709.4.1 template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator* ( ) [inline]

Simply returns *this.

Definition at line 542 of file stl_iterator.h.

4.709.4.2 template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator++ ( ) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 547 of file stl_iterator.h.

4.709.4.3 template<typename _Container > front_insert_iterator std::front_insert_iterator< _Container >::operator++ (int) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 552 of file stl_iterator.h.

4.709.4.4 template<typename _Container > front_insert_iterator& std::front_insert_iterator< _Container >::operator= (const typename _Container::value_type & __value) [inline]

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

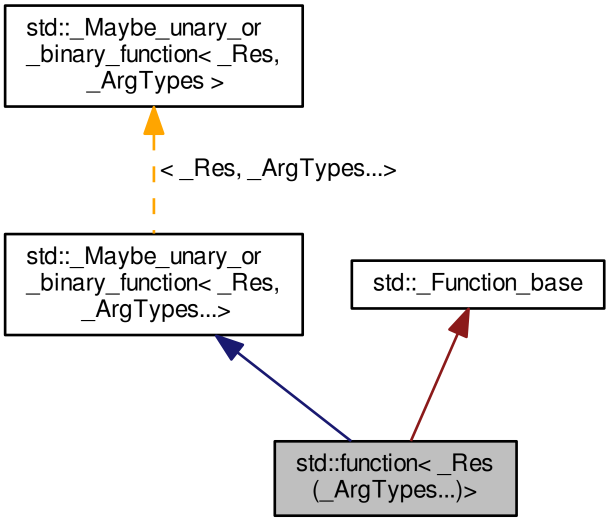
Definition at line 526 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- stl_iterator.h

4.710 std::function< _Res(_ArgTypes...)> Class Template Reference

Inheritance diagram for std::function< _Res(_ArgTypes...)>:



Public Types

- typedef _Res **result_type**

Public Member Functions

- function () noexcept
- function (nullptr_t) noexcept
- function (const function &__x)
- function (function &&__x)
- template<typename _Functor , typename = _Requires<_Callable<_Functor>, void>>
  function (_Functor)
- operator bool () const noexcept
- _Res operator() (_ArgTypes...__args) const
- function & operator= (const function &__x)
- function & operator= (function &&__x)
- function & operator= (nullptr_t)
- template<typename _Functor >
  _Requires< _Callable< _Functor >
  , function & > operator= (_Functor &&__f)
- template<typename _Functor >
  function & operator= (reference_wrapper< _Functor > __f) noexcept
- void swap (function &__x)

- template<typename _Functor >
  _Functor ∗ target () noexcept
- template<typename _Functor >
  const _Functor ∗ target () const noexcept
- const type_info & target_type () const noexcept

Private Types

- typedef bool(∗ **_Manager_type**)(_Any_data &, const _Any_data &, _Manager_operation)

Private Member Functions

- bool **_M_empty** () const

Private Attributes

- _Any_data **_M_functor**
- _Manager_type **_M_manager**

Static Private Attributes

- static const std::size_t **_M_max_align**
- static const std::size_t **_M_max_size**

4.710.1 Detailed Description

template<typename _Res, typename... _ArgTypes>class std::function< _Res(_ArgTypes...)>

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 2170 of file functional.

4.710.2 Constructor & Destructor Documentation

4.710.2.1 template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function ( ) `[inline]`, `[noexcept]`

Default construct creates an empty function call wrapper.

Postcondition

! (bool)∗this

Definition at line 2203 of file functional.

4.710.2.2 template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function (nullptr_t) [inline], [noexcept]

Creates an empty function call wrapper.

Postcondition

!(bool)*this

Definition at line 2210 of file functional.

4.710.2.3 template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function (const function< _Res(_ArgTypes...)> & __x)

Function copy constructor.

Parameters

__x	A function object with identical call signature.

Postcondition

`bool(*this)` == bool(__x)

The newly-created function contains a copy of the target of *__x* (if it has one).

Definition at line 2434 of file functional.

4.710.2.4 template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::function (function< _Res(_ArgTypes...)> && __x) [inline]

Function move constructor.

Parameters

__x	A function object rvalue with identical call signature.

The newly-created function contains the target of *__x* (if it has one).

Definition at line 2230 of file functional.

4.710.2.5 template<typename _Res , typename... _ArgTypes> template<typename _Functor , typename > std::function< _Res(_ArgTypes...)>::function (_Functor __f)

Builds a function that targets a copy of the incoming function object.

Parameters

__f	A function object that is callable with parameters of type `T1`, `T2`, ..., `TN` and returns a value convertible to `Res`.

The newly-created function object will target a copy of *__f*. If *__f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If *__f* is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If *__f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2448 of file functional.

4.710.3 Member Function Documentation

4.710.3.1 template<typename _Res , typename... _ArgTypes> std::function< _Res(_ArgTypes...)>::operator bool ( ) const `[inline]`, `[explicit]`, `[noexcept]`

Determine if the function wrapper has a target.

Returns

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 2383 of file functional.

4.710.3.2 template<typename _Res , typename... _ArgTypes> _Res std::function< _Res(_ArgTypes...)>::operator() (_ArgTypes... *__args*) const

Invokes the function targeted by `*this`.

Returns

the result of the target.

Exceptions

bad_function_call	when !(bool)*this

The function call operator invokes the target function object stored by `this`.

Definition at line 2464 of file functional.

4.710.3.3 template<typename _Res , typename... _ArgTypes> function& std::function< _Res(_ArgTypes...)>::operator= (const function< _Res(_ArgTypes...)> & *__x*) `[inline]`

Function assignment operator.

Parameters

__x	A function with identical call signature.

Postcondition

(bool)*this == (bool)x

Returns

`*this`

The target of *__x* is copied to `*this`. If *__x* has no target, then `*this` will be empty.

If *__x* targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 2270 of file functional.

4.710.3.4 template<typename _Res , typename... _ArgTypes> function& std::function< _Res(_ArgTypes...)>::operator= (function< _Res(_ArgTypes...)> && __x) [inline]

Function move-assignment operator.

Parameters

__x	A function rvalue with identical call signature.

Returns

`*this`

The target of *__x* is moved to `*this`. If *__x* has no target, then `*this` will be empty.

If *__x* targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 2288 of file functional.

4.710.3.5 template<typename _Res , typename... _ArgTypes> function& std::function< _Res(_ArgTypes...)>::operator= (nullptr_t) [inline]

Function assignment to zero.

Postcondition

!(bool)*this

Returns

`*this`

The target of `*this` is deallocated, leaving it empty.

Definition at line 2302 of file functional.

4.710.3.6 template<typename _Res , typename... _ArgTypes> template<typename _Functor > _Requires<_Callable<_Functor>, function&> std::function< _Res(_ArgTypes...)>::operator= (_Functor && __f) [inline]

Function assignment to a new target.

Parameters

__f	A function object that is callable with parameters of type `T1`, `T2`, ..., `TN` and returns a value convertible to `Res`.

Returns

`*this`

This function object wrapper will target a copy of *__f*. If *__f* is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If *__f* is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If *__f* is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 2331 of file functional.

4.710.3.7 template<typename _Res , typename... _ArgTypes> template<typename _Functor > **function**& std::function< _Res(_ArgTypes...)>::operator= (**reference_wrapper**< _Functor > *__f*) `[inline]`, `[noexcept]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2340 of file functional.

4.710.3.8 template<typename _Res , typename... _ArgTypes> void std::function< _Res(_ArgTypes...)>::swap (**function**< _Res(_ArgTypes...)> & *__x*) `[inline]`

Swap the targets of two function objects.

Parameters

__x	A function with identical call signature.

Swap the targets of `this` function object and *__f*. This function will not throw an exception.

Definition at line 2355 of file functional.

4.710.3.9 template<typename _Res , typename... _ArgTypes> template<typename _Functor > _Functor ∗ std::function< _Res(_ArgTypes...)>::target ( ) `[noexcept]`

Access the stored target function object.

Returns

> Returns a pointer to the stored target function object, if `typeid(Functor)`.equals(target_type()); otherwise, a NULL pointer.

This function will not throw an exception.

Definition at line 2491 of file functional.

4.710.3.10 template<typename _Res , typename... _ArgTypes> template<typename _Functor > const _Functor ∗ std::function< _Res(_ArgTypes...)>::target ( ) const `[noexcept]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 2510 of file functional.

4.710.3.11 template<typename _Res , typename... _ArgTypes> const **type_info** & std::function< _Res(_ArgTypes...)>::target_type ( ) const `[noexcept]`

Determine the type of the target of this function object wrapper.

Returns

> the type identifier of the target function object, or `typeid(void)` if !(bool)∗this.

This function will not throw an exception.

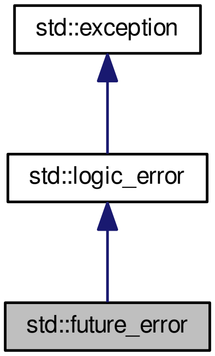
Definition at line 2475 of file functional.

The documentation for this class was generated from the following file:

- functional

4.711 std::future_error Class Reference

Inheritance diagram for std::future_error:



Public Member Functions

- **future_error** (error_code __ec)
- const error_code & **code** () const noexcept
- virtual const char ∗ what () const noexcept

4.711.1 Detailed Description

Exception type thrown by futures.

Definition at line 93 of file future.

4.711.2 Member Function Documentation

4.711.2.1 virtual const char∗ std::future_error::what ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

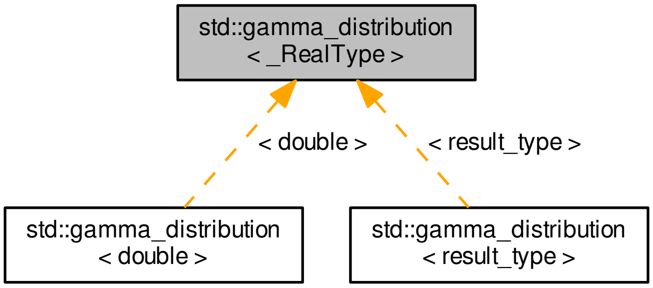
Reimplemented from std::logic_error.

The documentation for this class was generated from the following file:

- future

4.712 std::gamma_distribution< _RealType > Class Template Reference

Inheritance diagram for std::gamma_distribution< _RealType >:



Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- gamma_distribution (_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1))
- **gamma_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType alpha () const
- _RealType beta () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::gamma_distribution< _RealType1 > &__x)
- bool operator== (const gamma_distribution &__d1, const gamma_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::gamma_distribution< _RealType1 > &__x)

4.712.1 Detailed Description

template<typename _RealType = double>class std::gamma_distribution< _RealType >

A gamma continuous distribution for random numbers.

The formula for the gamma probability density function is:

$$p(x|\alpha,\beta) = \frac{1}{\beta\Gamma(\alpha)}(x/\beta)^{\alpha-1}e^{-x/\beta}$$

Definition at line 2502 of file random.h.

4.712.2 Member Typedef Documentation

4.712.2.1 template<typename _RealType = double> typedef _RealType std::gamma_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 2505 of file random.h.

4.712.3 Constructor & Destructor Documentation

4.712.3.1 template<typename _RealType = double> std::gamma_distribution< _RealType >::gamma_distribution (_RealType *__alpha_val* = `_RealType(1)`, _RealType *__beta_val* = `_RealType(1)`) `[inline]`, `[explicit]`

Constructs a gamma distribution with parameters α and β.

Definition at line 2554 of file random.h.

4.712.4 Member Function Documentation

4.712.4.1 template<typename _RealType = double> _RealType std::gamma_distribution< _RealType >::alpha ( ) const `[inline]`

Returns the α of the distribution.

Definition at line 2575 of file random.h.

4.712.4.2 template<typename _RealType = double> _RealType std::gamma_distribution< _RealType >::beta ( ) const [inline]

Returns the β of the distribution.

Definition at line 2582 of file random.h.

4.712.4.3 template<typename _RealType = double> result_type std::gamma_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 2611 of file random.h.

4.712.4.4 template<typename _RealType = double> result_type std::gamma_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 2604 of file random.h.

4.712.4.5 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::gamma_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 2619 of file random.h.

Referenced by std::gamma_distribution< result_type >::operator()().

4.712.4.6 template<typename _RealType > template<typename _UniformRandomNumberGenerator > gamma_distribution< _RealType >::result_type std::gamma_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*, const param_type & *__param*)

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2487 of file bits/random.tcc.

References std::log(), and std::pow().

4.712.4.7 template<typename _RealType = double> param_type std::gamma_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 2589 of file random.h.

4.712.4.8 template<typename _RealType = double> void std::gamma_distribution< _RealType >::param (const param_type & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 2597 of file random.h.

4.712.4.9 template<typename _RealType = double> void **std::gamma_distribution**< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 2568 of file random.h.

Referenced by std::chi_squared_distribution< _RealType >::reset(), std::fisher_f_distribution< _RealType >::reset(), std::student_t_distribution< _RealType >::reset(), and std::negative_binomial_distribution< _IntType >::reset().

4.712.5 Friends And Related Function Documentation

4.712.5.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > **std::basic_ostream**<_CharT, _Traits>& operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::gamma_distribution**< _RealType1 > & *__x*) [friend]

Inserts a gamma_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A gamma_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.712.5.2 template<typename _RealType = double> bool operator== (const **gamma_distribution**< _RealType > & *__d1*, const **gamma_distribution**< _RealType > & *__d2*) [friend]

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2655 of file random.h.

4.712.5.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > **std::basic_istream**<_CharT, _Traits>& operator>> (**std::basic_istream**< _CharT, _Traits > & *__is*, **std::gamma_distribution**< _RealType1 > & *__x*) [friend]

Extracts a gamma_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A gamma_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.713 std::gamma_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef gamma_distribution < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __alpha_val=_RealType(1), _RealType __beta_val=_RealType(1))
- _RealType **alpha** () const
- _RealType **beta** () const

Friends

- class **gamma_distribution< _RealType >**
- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.713.1 Detailed Description

template<typename _RealType = double>struct std::gamma_distribution< _RealType >::param_type

Parameter type.

Definition at line 2511 of file random.h.

The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

4.714 std::geometric_distribution< _IntType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **geometric_distribution** (double __p=0.5)
- **geometric_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator > void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator > void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- template< typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template< typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template< typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const geometric_distribution &__d1, const geometric_distribution &__d2)

4.714.1 Detailed Description

template< typename _IntType = int > class std::geometric_distribution< _IntType >

A discrete geometric random number distribution.

The formula for the geometric probability density function is $p(i|p) = p(1-p)^i$ where p is the parameter of the distribution.

Definition at line 4007 of file random.h.

4.714.2 Member Typedef Documentation

4.714.2.1 template< typename _IntType = int > typedef _IntType std::geometric_distribution< _IntType >::result_type

The type of the range of the distribution.

Definition at line 4010 of file random.h.

4.714.3 Member Function Documentation

4.714.3.1 template< typename _IntType = int > result_type std::geometric_distribution< _IntType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 4099 of file random.h.

References std::numeric_limits< _Tp >::max().

4.714.3.2 template< typename _IntType = int > result_type std::geometric_distribution< _IntType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4092 of file random.h.

4.714.3.3 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > **result_type std::geometric_distribution**< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 4107 of file random.h.

References std::geometric_distribution< _IntType >::operator()().

Referenced by std::geometric_distribution< _IntType >::operator()().

4.714.3.4 template<typename _IntType = int> double **std::geometric_distribution**< _IntType >::p ( ) const [inline]

Returns the distribution parameter `p`.

Definition at line 4070 of file random.h.

4.714.3.5 template<typename _IntType = int> **param_type std::geometric_distribution**< _IntType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 4077 of file random.h.

Referenced by std::operator>>().

4.714.3.6 template<typename _IntType = int> void **std::geometric_distribution**< _IntType >::param (const **param_type** & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 4085 of file random.h.

4.714.3.7 template<typename _IntType = int> void **std::geometric_distribution**< _IntType >::reset ( ) [inline]

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 4064 of file random.h.

4.714.4 Friends And Related Function Documentation

4.714.4.1 template<typename _IntType = int> bool operator== (const **geometric_distribution**< _IntType > & *__d1*, const **geometric_distribution**< _IntType > & *__d2*) [friend]

Return true if two geometric distributions have the same parameters.

Definition at line 4142 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.715 std::geometric_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef geometric_distribution
  < _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __p=0.5)
- double **p** () const

Friends

- class **geometric_distribution**< **_IntType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.715.1 Detailed Description

template<typename _IntType = int>struct std::geometric_distribution< _IntType >::param_type

Parameter type.

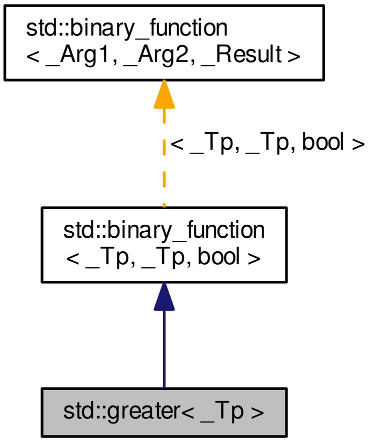
Definition at line 4016 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.716 std::greater< _Tp > Struct Template Reference

Inheritance diagram for std::greater< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.716.1 Detailed Description

template<typename _Tp>struct std::greater< _Tp >

One of the comparison functors.

Definition at line 222 of file stl_function.h.

4.716.2 Member Typedef Documentation

4.716.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.716.2.2 typedef bool **std::binary_function< _Tp , _Tp , bool >::result_type** [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.716.2.3 typedef _Tp **std::binary_function< _Tp , _Tp , bool >::second_argument_type** [inherited]

`second_argument_type` is the type of the second argument

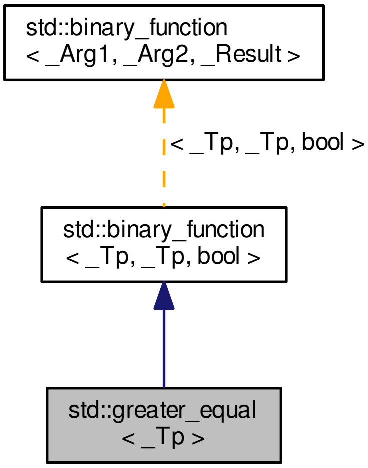
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.717 std::greater_equal< _Tp > Struct Template Reference

Inheritance diagram for std::greater_equal< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.717.1 Detailed Description

template<typename _Tp>struct std::greater_equal< _Tp >

One of the comparison functors.

Definition at line 240 of file stl_function.h.

4.717.2 Member Typedef Documentation

4.717.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.717.2.2 typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.717.2.3 typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.718 std::gslice Class Reference

Public Member Functions

- gslice ()
- gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- gslice (const gslice &)
- ~gslice ()
- gslice & operator= (const gslice &)
- valarray< size_t > size () const
- size_t start () const
- valarray< size_t > stride () const

Friends

- template<typename _Tp >
  class **valarray**

4.718.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have offset==3, stride[0]==11, size[1]==3, stride[1]==3, then slice[0,0]==array[3], slice[0,1]==array[6], slice[0,2]==array[9], slice[1,0]==array[14], slice[1,1]==array[17], slice[1,2]==array[20].

Definition at line 64 of file gslice.h.

The documentation for this class was generated from the following file:

- gslice.h

4.719 std::gslice_array< _Tp > Class Template Reference

Public Types

- typedef _Tp **value_type**

Public Member Functions

- gslice_array (const gslice_array &)
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- gslice_array & operator= (const gslice_array &)
- void operator= (const valarray< _Tp > &) const
- void operator= (const _Tp &) const
- template<class _Dom >
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const

- void operator^= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator|=** (const _Expr< _Dom, _Tp > &) const

Friends

- class **valarray< _Tp >**

4.719.1 Detailed Description

template<typename _Tp>class std::gslice_array< _Tp >

Reference to multi-dimensional subset of an array.

A gslice_array is a reference to the actual elements of an array specified by a gslice. The way to get a gslice_array is to call operator[](gslice) on a valarray. The returned gslice_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this gslice_array refers to.

Parameters

Tp	Element type.

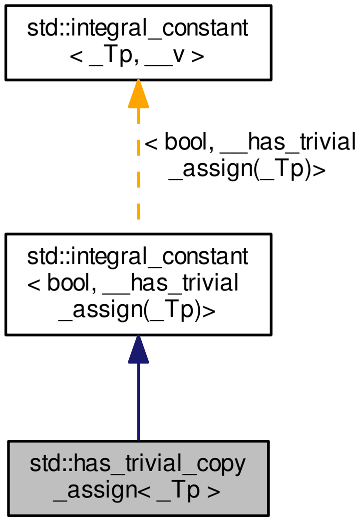
Definition at line 60 of file gslice_array.h.

The documentation for this class was generated from the following file:

- gslice_array.h

4.720 std::has_trivial_copy_assign< _Tp > Struct Template Reference

Inheritance diagram for std::has_trivial_copy_assign< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.720.1 Detailed Description

template<typename _Tp>struct std::has_trivial_copy_assign< _Tp >

has_trivial_copy_assign (temporary legacy)

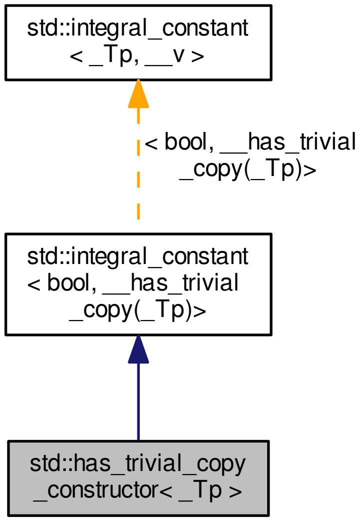
Definition at line 1223 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.721 std::has_trivial_copy_constructor< _Tp > Struct Template Reference

Inheritance diagram for std::has_trivial_copy_constructor< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.721.1 Detailed Description

template<typename _Tp>struct std::has_trivial_copy_constructor< _Tp >

has_trivial_copy_constructor (temporary legacy)

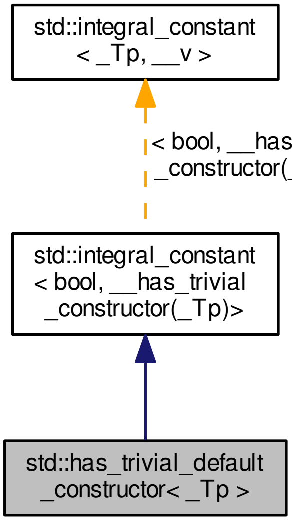
Definition at line 1217 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.722 std::has_trivial_default_constructor< _Tp > Struct Template Reference

Inheritance diagram for std::has_trivial_default_constructor< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.722.1 Detailed Description

template<typename _Tp>struct std::has_trivial_default_constructor< _Tp >

has_trivial_default_constructor (temporary legacy)

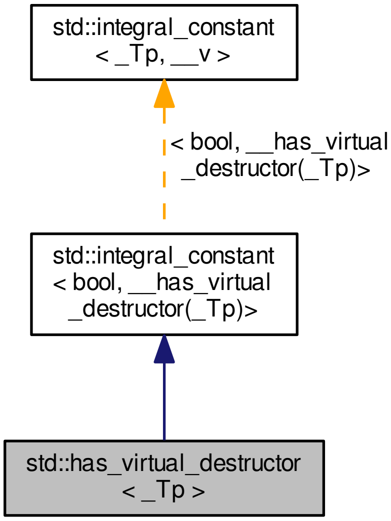
Definition at line 1211 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.723 std::has_virtual_destructor< _Tp > Struct Template Reference

Inheritance diagram for std::has_virtual_destructor< _Tp >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.723.1 Detailed Description

template<typename _Tp>struct std::has_virtual_destructor< _Tp >

has_virtual_destructor

Definition at line 1229 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.724 std::hash< __debug::bitset< _Nb > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __debug::bitset< _Nb > &__b) const noexcept

4.724.1 Detailed Description

template<size_t _Nb>struct std::hash< __debug::bitset< _Nb > >

std::hash specialization for bitset.

Definition at line 415 of file debug/bitset.

The documentation for this struct was generated from the following file:

- debug/bitset

4.725 std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __debug::vector< bool, _Alloc > &__b) const noexcept

4.725.1 Detailed Description

template<typename _Alloc>struct std::hash< __debug::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.

Definition at line 643 of file debug/vector.

The documentation for this struct was generated from the following file:

- debug/vector

4.726 std::hash< __gnu_cxx::__u16vstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__u16vstring &__s) const noexcept

4.726.1 Detailed Description

template<>struct std::hash< __gnu_cxx::__u16vstring >

std::hash specialization for __u16vstring.

Definition at line 2826 of file vstring.h.

The documentation for this struct was generated from the following file:

- vstring.h

4.727 std::hash< __gnu_cxx::__u32vstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__u32vstring &__s) const noexcept

4.727.1 Detailed Description

template<>struct std::hash< __gnu_cxx::__u32vstring >

std::hash specialization for __u32vstring.

Definition at line 2837 of file vstring.h.

The documentation for this struct was generated from the following file:

- vstring.h

4.728 std::hash< __gnu_cxx::__vstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__vstring &__s) const noexcept

4.728.1 Detailed Description

template<>struct std::hash< __gnu_cxx::__vstring >

std::hash specialization for __vstring.

Definition at line 2802 of file vstring.h.

The documentation for this struct was generated from the following file:

- vstring.h

4.729 std::hash< __gnu_cxx::__wvstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__wvstring &__s) const noexcept

4.729.1 Detailed Description

template<>struct std::hash< __gnu_cxx::__wvstring >

std::hash specialization for __wvstring.

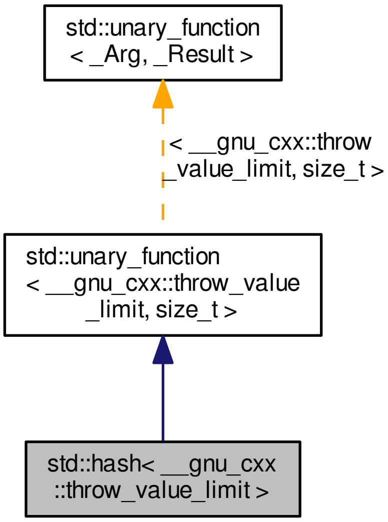
Definition at line 2813 of file vstring.h.

The documentation for this struct was generated from the following file:

- vstring.h

4.730 std::hash< __gnu_cxx::throw_value_limit > Struct Template Reference

Inheritance diagram for std::hash< __gnu_cxx::throw_value_limit >:



Public Types

- typedef __gnu_cxx::throw_value_limit argument_type
- typedef size_t result_type

Public Member Functions

- size_t **operator()** (const __gnu_cxx::throw_value_limit &__val) const

4.730.1 Detailed Description

template<>struct std::hash< __gnu_cxx::throw_value_limit >

Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.

Definition at line 788 of file throw_allocator.h.

4.730.2 Member Typedef Documentation

4.730.2.1 typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit , size_t >::argument_type `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.730.2.2 typedef size_t std::unary_function< __gnu_cxx::throw_value_limit , size_t >::result_type `[inherited]`

`result_type` is the return type

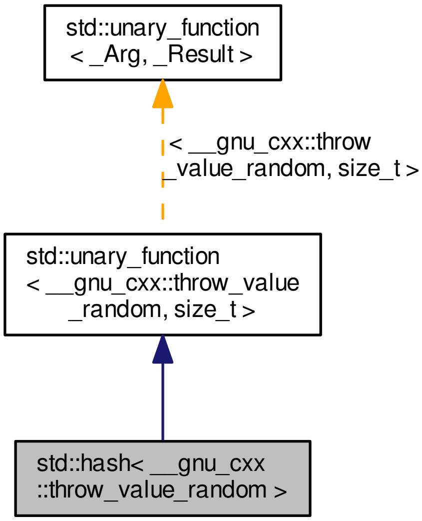
Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.731 std::hash< __gnu_cxx::throw_value_random > Struct Template Reference

Inheritance diagram for std::hash< __gnu_cxx::throw_value_random >:



Public Types

- typedef __gnu_cxx::throw_value_random argument_type
- typedef size_t result_type

Public Member Functions

- size_t **operator()** (const __gnu_cxx::throw_value_random &__val) const

4.731.1 Detailed Description

template<>struct std::hash< __gnu_cxx::throw_value_random >

Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.

Definition at line 802 of file throw_allocator.h.

4.731.2 Member Typedef Documentation

4.731.2.1 typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random , size_t >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.731.2.2 typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type [inherited]

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- throw_allocator.h

4.732 std::hash< __profile::bitset< _Nb > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __profile::bitset< _Nb > &__b) const noexcept

4.732.1 Detailed Description

template<size_t _Nb>struct std::hash< __profile::bitset< _Nb > >

std::hash specialization for bitset.

Definition at line 371 of file profile/bitset.

The documentation for this struct was generated from the following file:

- profile/bitset

4.733 std::hash< __profile::vector< bool, _Alloc > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __profile::vector< bool, _Alloc > &__b) const noexcept

4.733.1 Detailed Description

template<typename _Alloc>struct std::hash< __profile::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.

Definition at line 532 of file profile/vector.

The documentation for this struct was generated from the following file:

- profile/vector

4.734 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const __shared_ptr< _Tp, _Lp > &__s) const noexcept

4.734.1 Detailed Description

template<typename _Tp, _Lock_policy _Lp>struct std::hash< __shared_ptr< _Tp, _Lp > >

std::hash specialization for __shared_ptr.

Definition at line 1442 of file shared_ptr_base.h.

The documentation for this struct was generated from the following file:

- shared_ptr_base.h

4.735 std::hash< _Tp * > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (_Tp ∗__p) const noexcept

4.735.1 Detailed Description

template<typename _Tp>struct std::hash< _Tp ∗ >

Partial specializations for pointer types.

Definition at line 62 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.736 std::hash< bool > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (bool __val) const noexcept

4.736.1 Detailed Description

template<>struct std::hash< bool >

Explicit specialization for bool.

Definition at line 80 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.737 std::hash< char > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (char __val) const noexcept

4.737.1 Detailed Description

template<>struct std::hash< char >

Explicit specialization for char.

Definition at line 83 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.738 std::hash< char16_t > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (char16_t __val) const noexcept

4.738.1 Detailed Description

template<>struct std::hash< char16_t >

Explicit specialization for char16_t.

Definition at line 95 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.739 std::hash< char32_t > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (char32_t __val) const noexcept

4.739.1 Detailed Description

template<>struct std::hash< char32_t >

Explicit specialization for char32_t.

Definition at line 98 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.740 std::hash< double > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (double __val) const noexcept

4.740.1 Detailed Description

template<>struct std::hash< double >

Specialization for double.

Definition at line 176 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.741 std::hash< error_code > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const error_code &__e) const noexcept

4.741.1 Detailed Description

template<>struct std::hash< error_code >

std::hash specialization for error_code.

Definition at line 360 of file system_error.

The documentation for this struct was generated from the following file:

- system_error

4.742 std::hash< float > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (float __val) const noexcept

4.742.1 Detailed Description

template<>struct std::hash< float >

Specialization for float.

Definition at line 164 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.743 std::hash< int > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (int __val) const noexcept

4.743.1 Detailed Description

template<>struct std::hash< int >

Explicit specialization for int.

Definition at line 104 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.744 std::hash< long > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (long __val) const noexcept

4.744.1 Detailed Description

template<>struct std::hash< long >

Explicit specialization for long.

Definition at line 107 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.745 std::hash< long double > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (long double __val) const noexcept

4.745.1 Detailed Description

template<>struct std::hash< long double >

Specialization for long double.

Definition at line 188 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.746 std::hash< long long > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (long long __val) const noexcept

4.746.1 Detailed Description

template<>struct std::hash< long long >

Explicit specialization for long long.

Definition at line 110 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.747 std::hash< shared_ptr< _Tp > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const shared_ptr< _Tp > &__s) const noexcept

4.747.1 Detailed Description

template<typename _Tp>struct std::hash< shared_ptr< _Tp > >

std::hash specialization for shared_ptr.

Definition at line 619 of file shared_ptr.h.

The documentation for this struct was generated from the following file:

- shared_ptr.h

4.748 std::hash< short > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (short __val) const noexcept

4.748.1 Detailed Description

template<>struct std::hash< short >

Explicit specialization for short.

Definition at line 101 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.749 std::hash< signed char > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (signed char __val) const noexcept

4.749.1 Detailed Description

template<>struct std::hash< signed char >

Explicit specialization for signed char.

Definition at line 86 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.750 std::hash< string > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const string &__s) const noexcept

4.750.1 Detailed Description

template<>struct std::hash< string >

std::hash specialization for string.

Definition at line 3046 of file basic_string.h.

The documentation for this struct was generated from the following file:

- basic_string.h

4.751 std::hash< thread::id > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const thread::id &__id) const noexcept

4.751.1 Detailed Description

template<>struct std::hash< thread::id >

std::hash specialization for thread::id.

Definition at line 222 of file thread.

The documentation for this struct was generated from the following file:

- thread

4.752 std::hash< type_index > Struct Template Reference

Public Types

- typedef type_index **argument_type**
- typedef size_t **result_type**

Public Member Functions

- size_t **operator()** (const type_index &__ti) const noexcept

4.752.1 Detailed Description

template<>struct std::hash< type_index >

std::hash specialization for type_index.

Definition at line 97 of file typeindex.

The documentation for this struct was generated from the following file:

- typeindex

4.753 std::hash< u16string > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const u16string &__s) const noexcept

4.753.1 Detailed Description

template<>struct std::hash< u16string >

std::hash specialization for u16string.

Definition at line 3079 of file basic_string.h.

The documentation for this struct was generated from the following file:

- basic_string.h

4.754 std::hash< u32string > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const u32string &__s) const noexcept

4.754.1 Detailed Description

template<>struct std::hash< u32string >

std::hash specialization for u32string.

Definition at line 3094 of file basic_string.h.

The documentation for this struct was generated from the following file:

- basic_string.h

4.755 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const unique_ptr< _Tp, _Dp > &__u) const noexcept

4.755.1 Detailed Description

template<typename _Tp, typename _Dp>struct std::hash< unique_ptr< _Tp, _Dp > >

std::hash specialization for unique_ptr.

Definition at line 599 of file unique_ptr.h.

The documentation for this struct was generated from the following file:

- unique_ptr.h

4.756 std::hash< unsigned char > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (unsigned char __val) const noexcept

4.756.1 Detailed Description

template<>struct std::hash< unsigned char >

Explicit specialization for unsigned char.

Definition at line 89 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.757 std::hash< unsigned int > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (unsigned int __val) const noexcept

4.757.1 Detailed Description

template<>struct std::hash< unsigned int >

Explicit specialization for unsigned int.

Definition at line 116 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.758 std::hash< unsigned long > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (unsigned long __val) const noexcept

4.758.1 Detailed Description

template<>struct std::hash< unsigned long >

Explicit specialization for unsigned long.

Definition at line 119 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.759 std::hash< unsigned long long > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (unsigned long long __val) const noexcept

4.759.1 Detailed Description

template<>struct std::hash< unsigned long long >

Explicit specialization for unsigned long long.

Definition at line 122 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.760 std::hash< unsigned short > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (unsigned short __val) const noexcept

4.760.1 Detailed Description

template<>struct std::hash< unsigned short >

Explicit specialization for unsigned short.

Definition at line 113 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.761 std::hash< wchar_t > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (wchar_t __val) const noexcept

4.761.1 Detailed Description

template<>struct std::hash< wchar_t >

Explicit specialization for wchar_t.

Definition at line 92 of file functional_hash.h.

The documentation for this struct was generated from the following file:

- functional_hash.h

4.762 std::hash< wstring > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const wstring &__s) const noexcept

4.762.1 Detailed Description

template<>struct std::hash< wstring >

std::hash specialization for wstring.

Definition at line 3061 of file basic_string.h.

The documentation for this struct was generated from the following file:

- basic_string.h

4.763 std::hash<::bitset< _Nb > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const ::bitset< _Nb > &__b) const noexcept

4.763.1 Detailed Description

template<size_t _Nb>struct std::hash<::bitset< _Nb > >

std::hash specialization for bitset.

Definition at line 1552 of file bitset.

The documentation for this struct was generated from the following file:

- bitset

4.764 std::hash<::vector< bool, _Alloc > > Struct Template Reference

Inherits std::__hash_base< _Result, _Arg >.

Public Types

- typedef _Arg **argument_type**
- typedef _Result **result_type**

Public Member Functions

- size_t **operator()** (const ::vector< bool, _Alloc > &) const noexcept

4.764.1 Detailed Description

template<typename _Alloc>struct std::hash<::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.

Definition at line 1143 of file stl_bvector.h.

The documentation for this struct was generated from the following files:

- stl_bvector.h
- vector.tcc

4.765 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference

Public Types

- typedef _UIntType result_type

Public Member Functions

- independent_bits_engine ()
- independent_bits_engine (const _RandomNumberEngine &__rng)
- independent_bits_engine (_RandomNumberEngine &&__rng)
- independent_bits_engine (result_type __s)

- template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>
  independent_bits_engine (_Sseq &__q)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- void seed (result_type __s)
- template<typename _Sseq >
  void seed (_Sseq &__q)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Friends

- bool operator== (const independent_bits_engine &__lhs, const independent_bits_engine &__rhs)
- template<typename _CharT , typename _Traits >
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

4.765.1 Detailed Description

template<typename _RandomNumberEngine, size_t __w, typename _UIntType>class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1074 of file random.h.

4.765.2 Member Typedef Documentation

4.765.2.1 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type

The type of the generated random value.

Definition at line 1077 of file random.h.

4.765.3 Constructor & Destructor Documentation

4.765.3.1 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ( ) `[inline]`

Constructs a default independent_bits_engine engine.

The underlying engine is default constructed as well.

Definition at line 1090 of file random.h.

4.765.3.2 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (const _RandomNumberEngine & *__rng*) [inline], [explicit]

Copy constructs a independent_bits_engine engine.

Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

Definition at line 1100 of file random.h.

4.765.3.3 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (_RandomNumberEngine && *__rng*) [inline], [explicit]

Move constructs a independent_bits_engine engine.

Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

Definition at line 1110 of file random.h.

4.765.3.4 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (result_type *__s*) [inline], [explicit]

Seed constructs a independent_bits_engine engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

__s	A seed value for the base class engine.

Definition at line 1120 of file random.h.

4.765.3.5 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, independent_bits_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (_Sseq & *__q*) [inline], [explicit]

Generator construct a independent_bits_engine engine.

Parameters

__q	A seed sequence.

Definition at line 1133 of file random.h.

4.765.4 Member Function Documentation

4.765.4.1 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> const _RandomNumberEngine& std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::base ( ) const [inline], [noexcept]

Gets a const reference to the underlying generator engine object.

Definition at line 1168 of file random.h.

Referenced by std::operator<<().

4.765.4.2 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard (unsigned long long __z) [inline]

Discard a sequence of random numbers.

Definition at line 1189 of file random.h.

4.765.4.3 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::max ( ) [inline], [static]

Gets the maximum value in the generated random number range.

Definition at line 1182 of file random.h.

4.765.4.4 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> static constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::min ( ) [inline], [static]

Gets the minimum value in the generated random number range.

Definition at line 1175 of file random.h.

4.765.4.5 template<typename _RandomNumberEngine , size_t __w, typename _UIntType > independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator() ( )

Gets the next value in the generated random number sequence.

Definition at line 745 of file bits/random.tcc.

References std::__lg(), std::numeric_limits< _Tp >::max(), and std::numeric_limits< _Tp >::min().

4.765.4.6 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed ( ) [inline]

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

Definition at line 1142 of file random.h.

4.765.4.7 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (result_type __s) [inline]

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

Definition at line 1150 of file random.h.

4.765.4.8 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _Sseq > void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (_Sseq & __q) [inline]

Reseeds the independent_bits_engine object with the given seed sequence.

Parameters

__q	A seed generator function.

Definition at line 1160 of file random.h.

4.765.5 Friends And Related Function Documentation

4.765.5.1 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> bool operator== (const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & *__lhs*, const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & *__rhs*) `[friend]`

Compares two independent_bits_engine random number generator objects of the same type for equality.

Parameters

__lhs	A independent_bits_engine random number generator object.
__rhs	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1214 of file random.h.

4.765.5.2 template<typename _RandomNumberEngine, size_t __w, typename _UIntType> template<typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & *__is*, std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & *__x*) `[friend]`

Extracts the current state of a % subtract_with_carry_engine random number generator engine ___`x` from the input stream ___`is`.

Parameters

__is	An input stream.
__x	A independent_bits_engine random number generator engine.

Returns

The input stream with the state of ___`x` extracted or in an error state.

Definition at line 1232 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.766 std::indirect_array< _Tp > Class Template Reference

Public Types

- typedef _Tp **value_type**

Public Member Functions

- indirect_array (const indirect_array &)
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- indirect_array & operator= (const indirect_array &)
- void operator= (const valarray< _Tp > &) const
- void operator= (const _Tp &) const
- template<class _Dom >
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator|=** (const _Expr< _Dom, _Tp > &) const

Friends

- class **gslice_array< _Tp >**
- class **valarray< _Tp >**

4.766.1 Detailed Description

template<class _Tp>class std::indirect_array< _Tp >

Reference to arbitrary subset of an array.

An indirect_array is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an indirect_array is to call operator[](valarray<size_t>) on a valarray. The returned indirect_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if an indirect_array is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have array[0]==3, array[2]==2, and array[4]==1.

Parameters

Tp	Element type.

Definition at line 62 of file indirect_array.h.

The documentation for this class was generated from the following file:

- indirect_array.h

4.767 std::initializer_list< _E > Class Template Reference

Public Types

- typedef const _E ∗ **const_iterator**
- typedef const _E & **const_reference**
- typedef const _E ∗ **iterator**
- typedef const _E & **reference**
- typedef size_t **size_type**
- typedef _E **value_type**

Public Member Functions

- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **size** () const noexcept

4.767.1 Detailed Description

template<class _E>class std::initializer_list< _E >

initializer_list

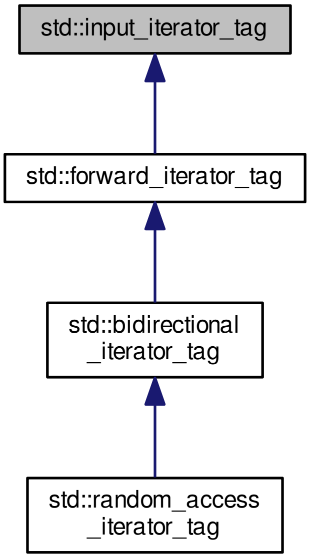
Definition at line 47 of file initializer_list.

The documentation for this class was generated from the following file:

- initializer_list

4.768 std::input_iterator_tag Struct Reference

Inheritance diagram for std::input_iterator_tag:



4.768.1 Detailed Description

Marking input iterators.

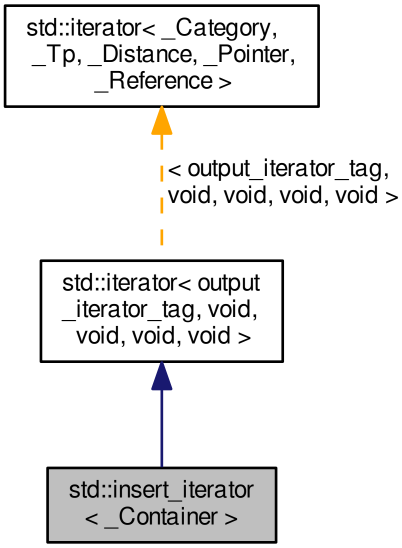
Definition at line 89 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.769 std::insert_iterator< _Container > Class Template Reference

Inheritance diagram for std::insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- insert_iterator (_Container &__x, typename _Container::iterator __i)
- insert_iterator & operator* ()
- insert_iterator & operator++ ()
- insert_iterator & operator++ (int)
- insert_iterator & operator= (const typename _Container::value_type &__value)
- insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**
- _Container::iterator **iter**

4.769.1 Detailed Description

template<typename _Container>class std::insert_iterator< _Container >

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 587 of file stl_iterator.h.

4.769.2 Member Typedef Documentation

4.769.2.1 template<typename _Container> typedef _Container **std::insert_iterator< _Container >::container_type**

A nested typedef for the type of whatever container you used.

Definition at line 596 of file stl_iterator.h.

4.769.2.2 typedef void **std::iterator< output_iterator_tag** , void , void , void , void **>::difference_type** `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.769.2.3 typedef **output_iterator_tag std::iterator< output_iterator_tag** , void , void , void , void **>::iterator_category** `[inherited]`

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.769.2.4 typedef void **std::iterator< output_iterator_tag** , void , void , void , void **>::pointer** `[inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.769.2.5 typedef void **std::iterator< output_iterator_tag** , void , void , void , void **>::reference** `[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.769.2.6 typedef void **std::iterator< output_iterator_tag** , void , void , void , void **>::value_type** `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.769.3 Constructor & Destructor Documentation

4.769.3.1 template<typename _Container> std::insert_iterator< _Container >::insert_iterator (_Container & *__x*, typename _Container::iterator *__i*) [inline]

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 602 of file stl_iterator.h.

4.769.4 Member Function Documentation

4.769.4.1 template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator* ( ) [inline]

Simply returns *this.

Definition at line 656 of file stl_iterator.h.

4.769.4.2 template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator++ ( ) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 661 of file stl_iterator.h.

4.769.4.3 template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator++ (int) [inline]

Simply returns *this. (This iterator does not *move*.)

Definition at line 666 of file stl_iterator.h.

4.769.4.4 template<typename _Container> insert_iterator& std::insert_iterator< _Container >::operator= (const typename _Container::value_type & *__value*) [inline]

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z

insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

Definition at line 638 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- stl_iterator.h

4.770 std::integral_constant< _Tp, __v > Struct Template Reference

Inherited by std::__and_<>, std::__is_base_to_derived_ref< _From, _To, false >, std::__is_copy_assignable_impl< _Tp, true >, std::__is_copy_constructible_impl< _Tp, true >, std::__is_destructible_safe< _Tp, false, true >, std::__is_destructible_safe< _Tp, true, false >, std::__is_empty_non_tuple< tuple< _El0, _El...> >, std::__is_fast_hash< _Hash >, std::__is_fast_hash< hash< long double > >, std::__is_fast_hash< hash< string > >, std::__is_fast_hash< hash< u16string > >, std::__is_fast_hash< hash< u32string > >, std::__is_fast_hash< hash< wstring > >, std::__is_floating_point_helper< typename >, std::__is_floating_point_helper< double >, std::__is_floating_point_helper< float >, std::__is_floating_point_helper< long double >, std::__is_integral_helper< typename >, std::__is_integral_helper< bool >, std::__is_integral_helper< char >, std::__is_integral_helper< char16_t >, std::__is_integral_helper< char32_t >, std::__is_integral_helper< int >, std::__is_integral_helper< long >, std::__is_integral_helper< long long >, std::__is_integral_helper< short >, std::__is_integral_helper< signed char >, std::__is_integral_helper< unsigned char >, std::__is_integral_helper< unsigned int >, std::__is_integral_helper< unsigned long >, std::__is_integral_helper< unsigned long long >, std::__is_integral_helper< unsigned short >, std::__is_integral_helper< wchar_t >, std::__is_lvalue_to_rvalue_ref< _From, _To, false >, std::__is_member_function_pointer_helper< typename >, std::__is_member_object_pointer_helper< typename >, std::__is_member_pointer_helper< _Tp >, std::__is_member_pointer_helper< _Tp _Cp::∗ >, std::__is_move_assignable_impl< _Tp, true >, std::__is_move_constructible_impl< _Tp, true >, std::__is_nothrow_copy_constructible_impl< _Tp, true >, std::__is_nothrow_move_constructible_impl< _Tp, true >, std::__is_nt_copy_assignable_impl< _Tp, true >, std::__is_nt_destructible_safe< _Tp, false, true >, std::__is_nt_destructible_safe< _Tp, true, false >, std::__is_nt_move_assignable_impl< _Tp, true >, std::__is_nullptr_t_helper< typename >, std::__is_nullptr_t_helper< std::nullptr_t >, std::__is_pointer_helper< typename >, std::__is_pointer_helper< _Tp ∗ >, std::__is_signed_helper< _Tp, bool, bool >, std::__is_signed_helper< _Tp, false, true >, std::__is_tuple_like< _Tp >, std::__is_tuple_like_impl< typename >, std::__is_tuple_like_impl< array< _Tp, _Nm > >, std::__is_tuple_like_impl< pair< _T1, _T2 > >, std::__is_tuple_like_impl< tuple< _Tps...> >, std::__is_void_helper< typename >, std::__is_void_helper< void >, std::__or_<>, std::__ratio_less_impl< _R1, _R2, bool, bool >, std::__uses_allocator_helper< _Tp, _Alloc, bool >, std::_AllConvertible< _From, _To, bool >, std::chrono::__is_duration< _Tp >, std::chrono::__is_duration< duration< _Rep, _Period > >, std::chrono::__is_ratio< T >, std::chrono::__is_ratio< ratio< _Num, _Den > >, std::function< _Res(_ArgTypes...)>::_CheckResult< _CallRes, void >, std::is_array< typename >, std::is_array< _Tp[]>, std::is_array< _Tp[_Size]>, std::is_bind_expression< _Tp >, std::is_bind_expression< _Bind< _Signature > >, std::is_bind_expression< _Bind_result< _Result, _Signature > >, std::is_bind_expression< const _Bind< _Signature > >, std::is_bind_expression< const _Bind_result< _Result, _Signature > >, std::is_bind_expression< const volatile _Bind< _Signature > >, std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >, std::is_bind_expression< volatile _Bind< _Signature > >, std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >, std::is_const< typename >, std::is_const< _Tp const >, std::is_error_code_enum< _Tp >, std::is_error_code_enum< future_errc >, std::is_error_condition_enum< _Tp >, std::is_error_condition_enum< errc >, std::is_function< typename >, std::is_function< _Res(_ArgTypes...) const >, std::is_function< _Res(_ArgTypes...) const volatile >, std::is_function< _Res(_ArgTypes...) volatile >, std::is_function< _Res(_ArgTypes...)>, std::is_function< _Res(_ArgTypes......) const >, std::is_function< _Res(_ArgTypes......) const volatile >, std::is_function< _Res(_ArgTypes......) volatile >, std::is_function< _Res(_ArgTypes......)>, std::is_lvalue_reference< typename >, std::is_lvalue_reference< _Tp & >, std::is_object< _Tp >, std::is_rvalue_reference< typename >, std::is_rvalue_reference< _Tp && >, std::is_same< typename, typename >, std::is_same< _Tp, _Tp >, std::is_volatile< typename >, std::is_volatile< _Tp volatile >, std::uses_allocator< priority_queue< _Tp, _Sequence, _Compare >, _Alloc >, std::uses_allocator< queue< _Tp, _Seq >, _Alloc >, std::uses_allocator< stack< _Tp, _Seq >, _Alloc >, and std::uses_allocator< tuple< _Types...>, _Alloc >.

Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.770.1 Detailed Description

template<typename _Tp, _Tp __v>struct std::integral_constant< _Tp, __v >

integral_constant

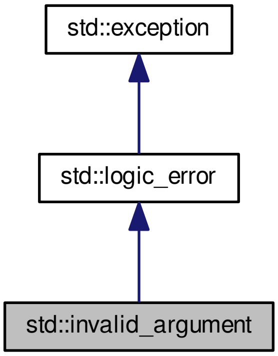
Definition at line 57 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.771 std::invalid_argument Class Reference

Inheritance diagram for std::invalid_argument:



Public Member Functions

- **invalid_argument** (const string &__arg)
- virtual const char $*$ what () const noexcept

4.771.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 82 of file stdexcept.

4.771.2 Member Function Documentation

4.771.2.1 virtual const char∗ std::logic_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

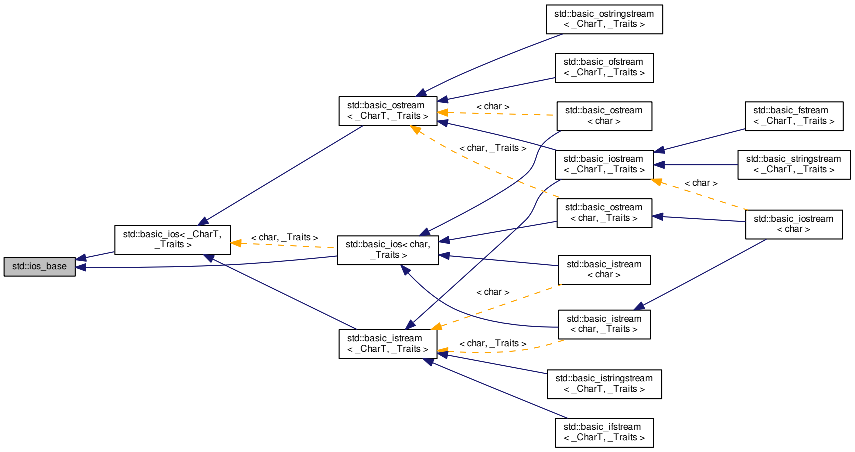
Reimplemented in std::future_error.

The documentation for this class was generated from the following file:

- stdexcept

4.772 std::ios_base Class Reference

Inheritance diagram for std::ios_base:



Classes

- class failure

  These are thrown to indicate problems with io.
  27.4.2.1.1 Class ios_base::failure.

Public Types

- enum event { **erase_event**, **imbue_event**, **copyfmt_event** }
- typedef void(∗ event_callback)(event __e, ios_base &__b, int __i)

- typedef _Ios_Fmtflags fmtflags
- typedef int **io_state**
- typedef _Ios_Iostate iostate
- typedef int **open_mode**
- typedef _Ios_Openmode openmode
- typedef int **seek_dir**
- typedef _Ios_Seekdir seekdir
- typedef std::streamoff **streamoff**
- typedef std::streampos **streampos**

Public Member Functions

- virtual ∼ios_base ()
- const locale & _M_getloc () const
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- locale getloc () const
- locale imbue (const locale &__loc) throw ()
- long & iword (int __ix)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void ∗& pword (int __ix)
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void unsetf (fmtflags __mask)
- streamsize width () const
- streamsize width (streamsize __wide)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed

- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()

Protected Attributes

- _Callback_list ∗ **_M_callbacks**
- iostate **_M_exception**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- streamsize **_M_precision**
- iostate **_M_streambuf_state**
- streamsize **_M_width**
- _Words ∗ **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

4.772.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 199 of file ios_base.h.

4.772.2 Member Typedef Documentation

4.772.2.1 typedef void(∗ std::ios_base::event_callback)(event __e, ios_base &__b, int __i)

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

Definition at line 436 of file ios_base.h.

4.772.2.2 typedef _Ios_Fmtflags std::ios_base::fmtflags

This is a bitmask type.

`_Ios_Fmtflags is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 255 of file ios_base.h.

4.772.2.3 typedef _Ios_Iostate std::ios_base::iostate

This is a bitmask type.

`_Ios_Iostate is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 330 of file ios_base.h.

4.772.2.4 typedef _Ios_Openmode std::ios_base::openmode

This is a bitmask type.

`_Ios_Openmode is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 361 of file ios_base.h.

4.772.2.5 typedef _Ios_Seekdir std::ios_base::seekdir

This is an enumerated type.

`_Ios_Seekdir is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

Definition at line 393 of file ios_base.h.

4.772.3 Member Enumeration Documentation

4.772.3.1 enum std::ios_base::event

The set of events that may be passed to an event callback.

erase_event is used during ∼ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

Definition at line 419 of file ios_base.h.

4.772.4 Constructor & Destructor Documentation

4.772.4.1 virtual std::ios_base::∼ios_base ( ) [virtual]

Invokes each callback with erase_event. Destroys local storage.

Note that the ios_base object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with erase_event (unless copyfmt is used).

4.772.5 Member Function Documentation

4.772.5.1 const locale& std::ios_base::_M_getloc ( ) const [inline]

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 706 of file ios_base.h.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::time_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), and std::time_put< _CharT, _OutIter >::put().

4.772.5.2 fmtflags std::ios_base::flags ( ) const [inline]

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 551 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::operator>>(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.772.5.3 fmtflags std::ios_base::flags (fmtflags *__fmtfl*) [inline]

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

Definition at line 562 of file ios_base.h.

4.772.5.4 **locale** std::ios_base::getloc ( ) const `[inline]`

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 695 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::ws().

4.772.5.5 **locale** std::ios_base::imbue (const **locale** & *__loc*) throw ()

Setting a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue_event.

Referenced by std::basic_ios< _CharT, _Traits >::imbue().

4.772.5.6 long& std::ios_base::iword (int *__ix*) `[inline]`

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 741 of file ios_base.h.

4.772.5.7 **streamsize** std::ios_base::precision ( ) const `[inline]`

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 621 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::operator<<().

4.772.5.8 streamsize std::ios_base::precision (streamsize *__prec*) [inline]

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

Definition at line 630 of file ios_base.h.

4.772.5.9 void∗& std::ios_base::pword (int *__ix*) [inline]

Access to void pointer array.

Parameters

__ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 762 of file ios_base.h.

4.772.5.10 void std::ios_base::register_callback (event_callback *__fn,* int *__index*)

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

4.772.5.11 fmtflags std::ios_base::setf (fmtflags *__fmtfl*) `[inline]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 578 of file ios_base.h.

Referenced by std::dec(), std::fixed(), std::hex(), std::left(), std::oct(), std::right(), std::scientific(), std::showbase(), std::showpoint(), std::showpos(), std::skipws(), std::unitbuf(), and std::uppercase().

4.772.5.12 fmtflags std::ios_base::setf (fmtflags *__fmtfl,* fmtflags *__mask*) `[inline]`

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 595 of file ios_base.h.

4.772.5.13 static bool std::ios_base::sync_with_stdio (bool *__sync* = `true`) `[static]`

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt1` `html`

4.772.5.14 void std::ios_base::unsetf (fmtflags *__mask*) `[inline]`

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

Definition at line 610 of file ios_base.h.

Referenced by std::noboolalpha(), std::noshowbase(), std::noshowpoint(), std::noshowpos(), std::noskipws(), std::nounitbuf(), and std::nouppercase().

4.772.5.15 streamsize std::ios_base::width ( ) const [inline]

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

Definition at line 644 of file ios_base.h.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), and std::operator>>().

4.772.5.16 streamsize std::ios_base::width (streamsize *__wide*) [inline]

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

Definition at line 653 of file ios_base.h.

4.772.5.17 static int std::ios_base::xalloc ( ) throw () [static]

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

4.772.6 Member Data Documentation

4.772.6.1 const **fmtflags** std::ios_base::adjustfield [static]

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 310 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

4.772.6.2 const **openmode** std::ios_base::app [static]

Seek to end before each write.

Definition at line 364 of file ios_base.h.

4.772.6.3 const **openmode** std::ios_base::ate [static]

Open and seek to end immediately after opening.

Definition at line 367 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open().

4.772.6.4 const **iostate** std::ios_base::badbit [static]

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 334 of file ios_base.h.

Referenced by std::basic_ostream< char >::_M_write(), std::basic_ios< char, _Traits >::bad(), std::basic_ios< char, _Traits >::fail(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tellp(), std::basic_istream< _CharT, _Traits >::unget(), std::basic_ostream< _CharT, _Traits >::write(), and std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

4.772.6.5 const **fmtflags** std::ios_base::basefield [static]

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 313 of file ios_base.h.

Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::hex(), std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.772.6.6 const **seekdir** std::ios_base::beg [static]

Request a seek relative to the beginning of the stream.

Definition at line 396 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

4.772.6.7 const **openmode** std::ios_base::binary [static]

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch27s02.html`.

Definition at line 372 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

4.772.6.8 const fmtflags std::ios_base::boolalpha [static]

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 258 of file ios_base.h.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

4.772.6.9 const seekdir std::ios_base::cur [static]

Request a seek relative to the current position within the sequence.

Definition at line 399 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

4.772.6.10 const fmtflags std::ios_base::dec [static]

Converts integer input or generates integer output in decimal base.

Definition at line 261 of file ios_base.h.

Referenced by std::dec().

4.772.6.11 const seekdir std::ios_base::end [static]

Request a seek relative to the current end of the sequence.

Definition at line 402 of file ios_base.h.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

4.772.6.12 const iostate std::ios_base::eofbit [static]

Indicates that an input operation reached the end of an input sequence.

Definition at line 337 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::eof(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

4.772.6.13 const iostate std::ios_base::failbit [static]

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 342 of file ios_base.h.

Referenced by std::basic_ifstream< _CharT, _Traits >::close(), std::basic_ofstream< _CharT, _Traits >::close(), std::basic_fstream< _CharT, _Traits >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ios< char, _Traits >::fail(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_fstream< _CharT, _Traits >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::basic_istream< _CharT, _Traits >::sentry::sentry().

4.772.6.14 const **fmtflags** std::ios_base::fixed `[static]`

Generate floating-point output in fixed-point notation.

Definition at line 264 of file ios_base.h.

Referenced by std::fixed().

4.772.6.15 const **fmtflags** std::ios_base::floatfield `[static]`

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 316 of file ios_base.h.

Referenced by std::fixed(), and std::scientific().

4.772.6.16 const **iostate** std::ios_base::goodbit `[static]`

Indicates all is well.

Definition at line 345 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::peek(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< _CharT, _Traits >::sync(), and std::basic_istream< _CharT, _Traits >::unget().

4.772.6.17 const **fmtflags** std::ios_base::hex `[static]`

Converts integer input or generates integer output in hexadecimal base.

Definition at line 267 of file ios_base.h.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.772.6.18 const **openmode** std::ios_base::in `[static]`

Open for input. Default for `ifstream` and fstream.

Definition at line 375 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ifstream< _CharT, _Traits >::open(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic-

_istream< _CharT, _Traits >::tellg(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

4.772.6.19 const **fmtflags** std::ios_base::internal `[static]`

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 272 of file ios_base.h.

Referenced by std::internal().

4.772.6.20 const **fmtflags** std::ios_base::left `[static]`

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 276 of file ios_base.h.

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

4.772.6.21 const **fmtflags** std::ios_base::oct `[static]`

Converts integer input or generates integer output in octal base.

Definition at line 279 of file ios_base.h.

Referenced by std::oct(), and std::basic_ostream< _CharT, _Traits >::operator<<().

4.772.6.22 const **openmode** std::ios_base::out `[static]`

Open for output. Default for `ofstream` and fstream.

Definition at line 378 of file ios_base.h.

Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_ofstream< _CharT, _Traits >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

4.772.6.23 const **fmtflags** std::ios_base::right `[static]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 283 of file ios_base.h.

Referenced by std::right().

4.772.6.24 const **fmtflags** std::ios_base::scientific `[static]`

Generates floating-point output in scientific notation.

Definition at line 286 of file ios_base.h.

Referenced by std::scientific().

4.772.6.25 const **fmtflags** std::ios_base::showbase `[static]`

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 290 of file ios_base.h.

Referenced by std::noshowbase(), and std::showbase().

4.772.6.26 const **fmtflags** std::ios_base::showpoint `[static]`

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 294 of file ios_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

4.772.6.27 const **fmtflags** std::ios_base::showpos `[static]`

Generates a + sign in non-negative generated numeric output.

Definition at line 297 of file ios_base.h.

Referenced by std::noshowpos(), and std::showpos().

4.772.6.28 const **fmtflags** std::ios_base::skipws `[static]`

Skips leading white space before certain input operations.

Definition at line 300 of file ios_base.h.

Referenced by std::noskipws(), std::basic_istream< _CharT, _Traits >::sentry::sentry(), and std::skipws().

4.772.6.29 const **openmode** std::ios_base::trunc `[static]`

Open for input. Default for `ofstream`.

Definition at line 381 of file ios_base.h.

4.772.6.30 const **fmtflags** std::ios_base::unitbuf `[static]`

Flushes output after each output operation.

Definition at line 303 of file ios_base.h.

Referenced by std::nounitbuf(), std::unitbuf(), and std::basic_ostream< _CharT, _Traits >::sentry::∼sentry().

4.772.6.31 const **fmtflags** std::ios_base::uppercase `[static]`

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 307 of file ios_base.h.

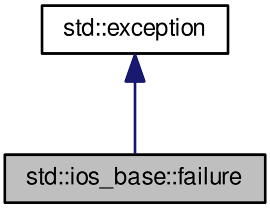
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- ios_base.h

4.773 std::ios_base::failure Class Reference

Inheritance diagram for std::ios_base::failure:



Public Member Functions

- **failure** (const string &__str) throw ()
- virtual const char ∗ what () const throw ()

4.773.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class ios_base::failure.

Definition at line 209 of file ios_base.h.

4.773.2 Member Function Documentation

4.773.2.1 virtual const char∗ std::ios_base::failure::what ( ) const throw () `[virtual]`

Returns a C-style character string describing the general cause of the current error.

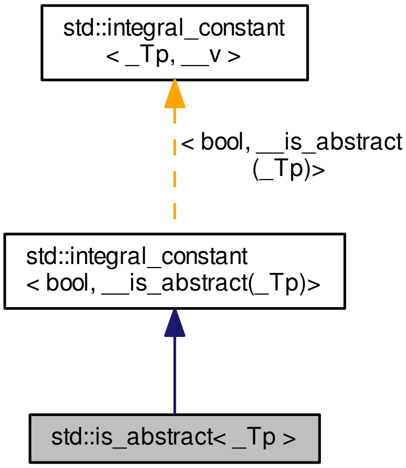
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- ios_base.h

4.774 std::is_abstract< _Tp > Struct Template Reference

Inheritance diagram for std::is_abstract< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.774.1 Detailed Description

template<typename _Tp>struct std::is_abstract< _Tp >

is_abstract

Definition at line 548 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.775 std::is_arithmetic< _Tp > Struct Template Reference

Inherits type< is_integral< _Tp >, is_floating_point< _Tp > >.

4.775.1 Detailed Description

template<typename _Tp>struct std::is_arithmetic< _Tp >

is_arithmetic

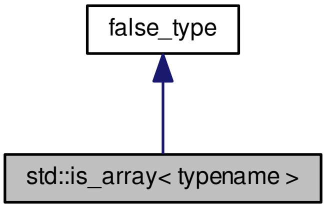
Definition at line 440 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.776 std::is_array< typename > Struct Template Reference

Inheritance diagram for std::is_array< typename >:



Public Types

- typedef integral_constant< _Tp,
  __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.776.1 Detailed Description

template<typename>struct std::is_array< typename >

is_array

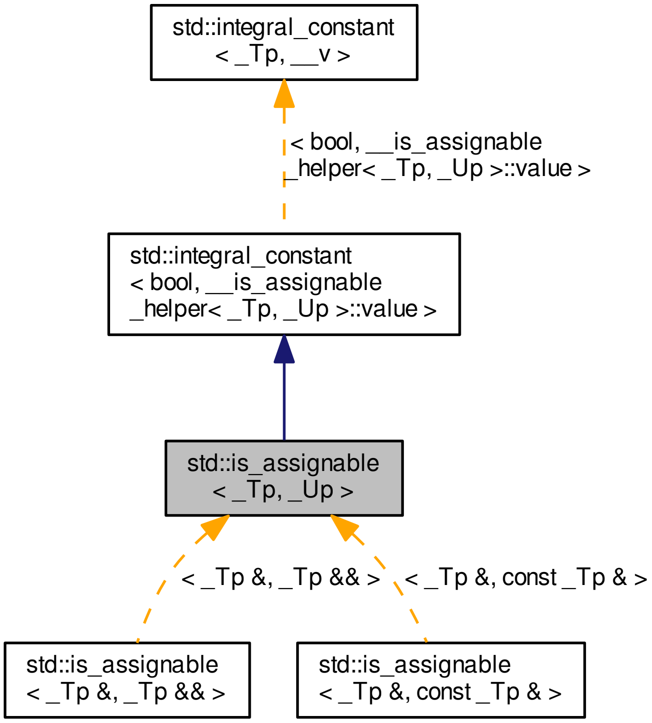
Definition at line 282 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.777 std::is_assignable< _Tp, _Up > Struct Template Reference

Inheritance diagram for std::is_assignable< _Tp, _Up >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.777.1 Detailed Description

template<typename _Tp, typename _Up>struct std::is_assignable< _Tp, _Up >

is_assignable

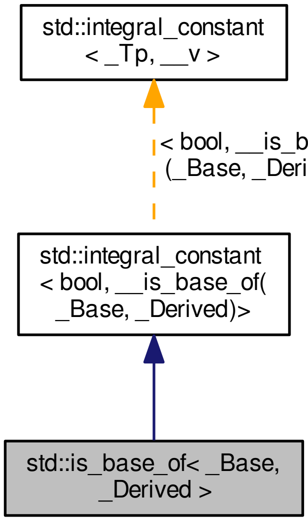
Definition at line 1099 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.778 std::is_base_of< _Base, _Derived > Struct Template Reference

Inheritance diagram for std::is_base_of< _Base, _Derived >:



Public Types

- typedef integral_constant
  < bool, __v > **type**

- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.778.1 Detailed Description

template<typename _Base, typename _Derived>struct std::is_base_of< _Base, _Derived >

is_base_of

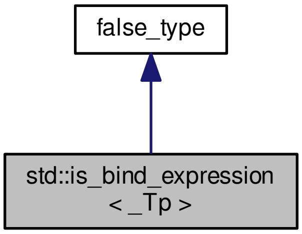
Definition at line 1287 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.779 std::is_bind_expression< _Tp > Struct Template Reference

Inheritance diagram for std::is_bind_expression< _Tp >:



Public Types

- typedef integral_constant< _Tp,
  __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.779.1 Detailed Description

template<typename _Tp>struct std::is_bind_expression< _Tp >

Determines if the given type _Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].

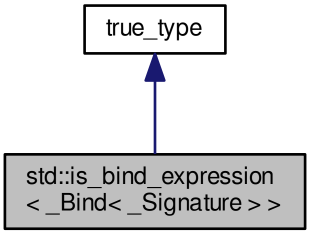
Definition at line 973 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.780 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< _Bind< _Signature > >:



Public Types

- typedef integral_constant< _Tp,
  __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.780.1 Detailed Description

template<typename _Signature>struct std::is_bind_expression< _Bind< _Signature > >

Class template _Bind is always a bind expression.

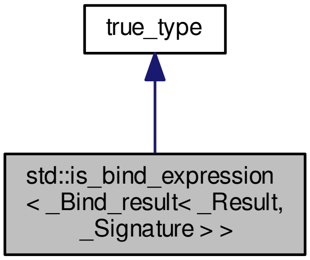
Definition at line 1567 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.781 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< _Bind_result< _Result, _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.781.1 Detailed Description

template<typename _Result, typename _Signature>struct std::is_bind_expression< _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.

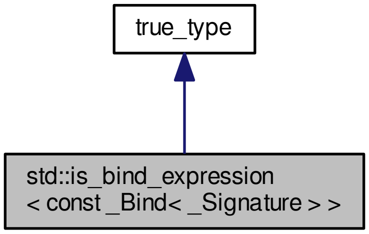
Definition at line 1599 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.782 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< const _Bind< _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.782.1 Detailed Description

template<typename _Signature>struct std::is_bind_expression< const _Bind< _Signature > >

Class template _Bind is always a bind expression.

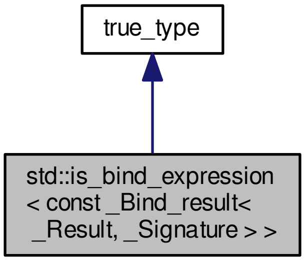
Definition at line 1575 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.783 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< const _Bind_result< _Result, _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.783.1 Detailed Description

template<typename _Result, typename _Signature>struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.

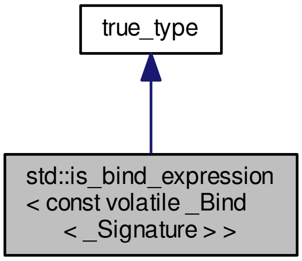
Definition at line 1607 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.784 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< const volatile _Bind< _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.784.1 Detailed Description

template<typename _Signature>struct std::is_bind_expression< const volatile _Bind< _Signature > >

Class template _Bind is always a bind expression.

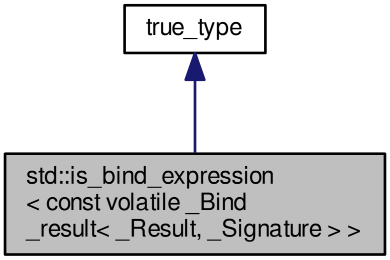
Definition at line 1591 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.785 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.785.1 Detailed Description

template<typename _Result, typename _Signature>struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.

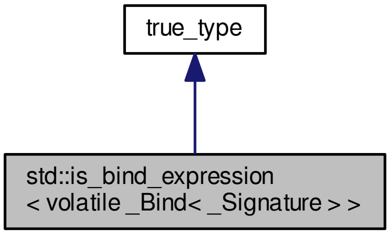
Definition at line 1623 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.786 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< volatile _Bind< _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.786.1 Detailed Description

template<typename _Signature>struct std::is_bind_expression< volatile _Bind< _Signature > >

Class template _Bind is always a bind expression.

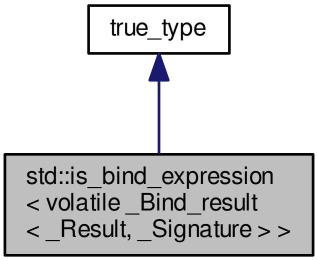
Definition at line 1583 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.787 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference

Inheritance diagram for std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.787.1 Detailed Description

template<typename _Result, typename _Signature>struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.

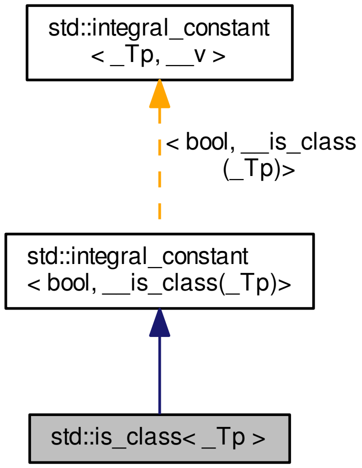
Definition at line 1615 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.788 std::is_class< _Tp > Struct Template Reference

Inheritance diagram for std::is_class< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.788.1 Detailed Description

template<typename _Tp>struct std::is_class< _Tp >

is_class

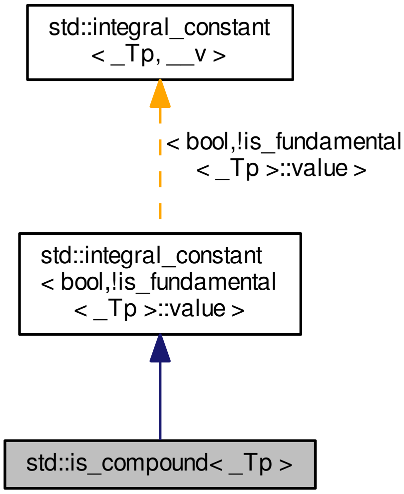
Definition at line 373 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.789 std::is_compound< _Tp > Struct Template Reference

Inheritance diagram for std::is_compound< _Tp >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.789.1 Detailed Description

template<typename _Tp>struct std::is_compound< _Tp >

is_compound

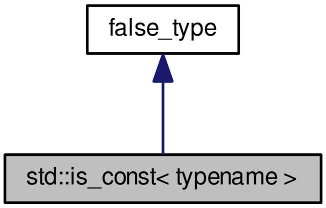
Definition at line 469 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.790 std::is_const< typename > Struct Template Reference

Inheritance diagram for std::is_const< typename >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.790.1 Detailed Description

template<typename>struct std::is_const< typename >

is_const

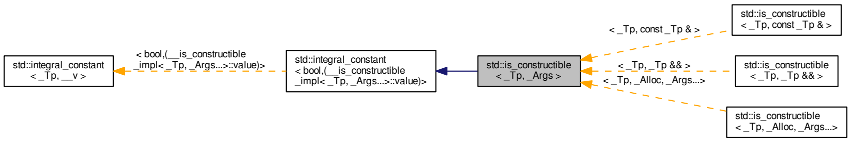
Definition at line 491 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.791 std::is_constructible< _Tp, _Args > Struct Template Reference

Inheritance diagram for std::is_constructible< _Tp, _Args >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.791.1 Detailed Description

template<typename _Tp, typename... _Args>struct std::is_constructible< _Tp, _Args >

is_constructible

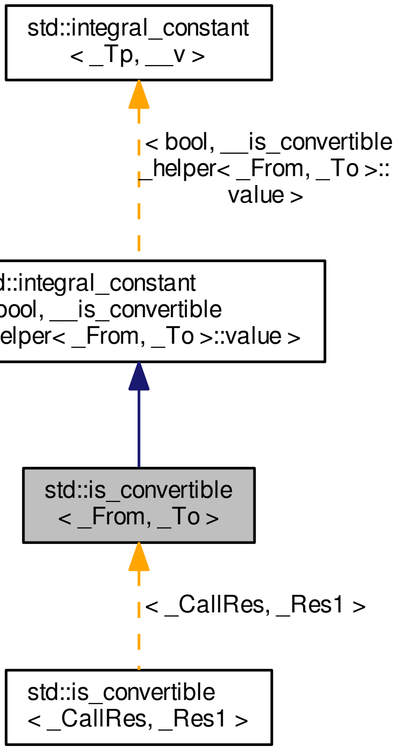
Definition at line 955 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.792 std::is_convertible< _From, _To > Struct Template Reference

Inheritance diagram for std::is_convertible< _From, _To >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.792.1 Detailed Description

template<typename _From, typename _To>struct std::is_convertible< _From, _To >

is_convertible

Definition at line 1317 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.793 std::is_copy_assignable< _Tp > Struct Template Reference

Inherits __is_copy_assignable_impl< _Tp, bool >.

4.793.1 Detailed Description

template<typename _Tp>struct std::is_copy_assignable< _Tp >

is_copy_assignable

Definition at line 1118 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.794 std::is_copy_constructible< _Tp > Struct Template Reference

Inherits __is_copy_constructible_impl< _Tp, bool >.

Inherited by std::__is_copy_insertable< allocator< _Tp > >.

4.794.1 Detailed Description

template<typename _Tp>struct std::is_copy_constructible< _Tp >

is_copy_constructible

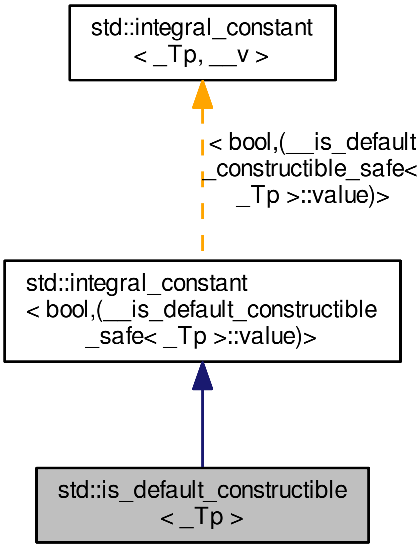
Definition at line 974 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.795 std::is_default_constructible< _Tp > Struct Template Reference

Inheritance diagram for std::is_default_constructible< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.795.1 Detailed Description

template<typename _Tp>struct std::is_default_constructible< _Tp >

is_default_constructible

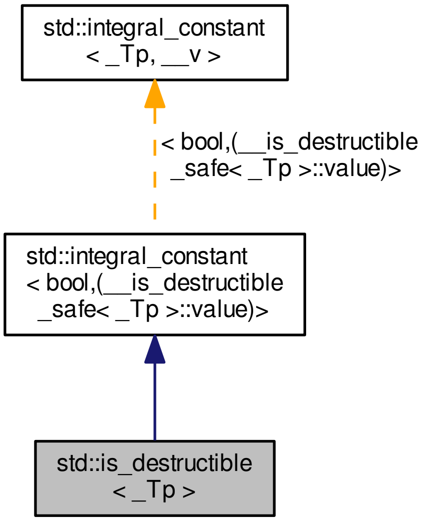
Definition at line 748 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.796 std::is_destructible< _Tp > Struct Template Reference

Inheritance diagram for std::is_destructible< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.796.1 Detailed Description

template<typename _Tp>struct std::is_destructible< _Tp >

is_destructible

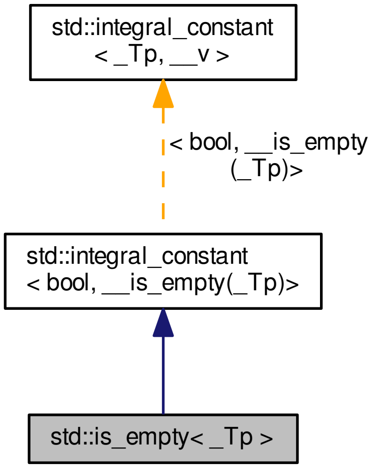
Definition at line 652 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.797 std::is_empty< _Tp > Struct Template Reference

Inheritance diagram for std::is_empty< _Tp >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.797.1 Detailed Description

template<typename _Tp>struct std::is_empty< _Tp >

is_empty

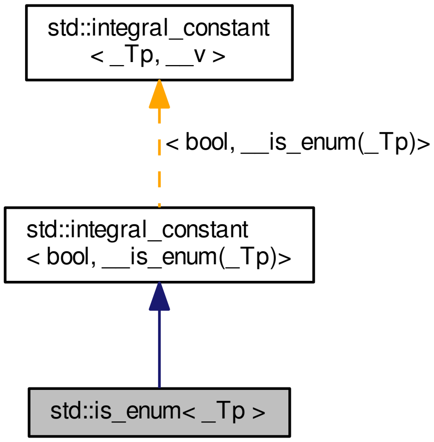
Definition at line 536 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.798 std::is_enum< _Tp > Struct Template Reference

Inheritance diagram for std::is_enum< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.798.1 Detailed Description

template<typename _Tp>struct std::is_enum< _Tp >

is_enum

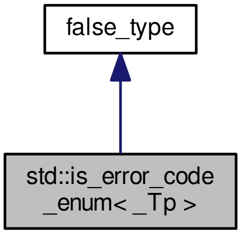
Definition at line 361 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.799 std::is_error_code_enum< _Tp > Struct Template Reference

Inheritance diagram for std::is_error_code_enum< _Tp >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.799.1 Detailed Description

template<typename _Tp>struct std::is_error_code_enum< _Tp >

is_error_code_enum

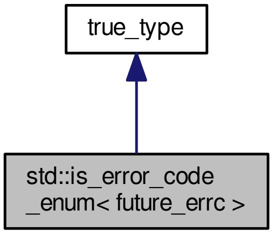
Definition at line 54 of file system_error.

The documentation for this struct was generated from the following file:

- system_error

4.800 std::is_error_code_enum< future_errc > Struct Template Reference

Inheritance diagram for std::is_error_code_enum< future_errc >:



Public Types

- typedef integral_constant< _Tp,
  __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.800.1 Detailed Description

template<>struct std::is_error_code_enum< future_errc >

Specialization.

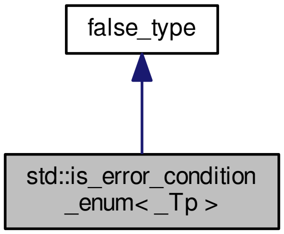
Definition at line 73 of file future.

The documentation for this struct was generated from the following file:

- future

4.801 std::is_error_condition_enum< _Tp > Struct Template Reference

Inheritance diagram for std::is_error_condition_enum< _Tp >:



Public Types

- typedef integral_constant< _Tp,
  __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.801.1 Detailed Description

template<typename _Tp>struct std::is_error_condition_enum< _Tp >

is_error_condition_enum

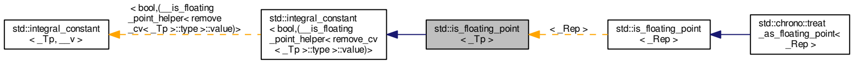
Definition at line 58 of file system_error.

The documentation for this struct was generated from the following file:

- system_error

4.802 std::is_floating_point< _Tp > Struct Template Reference

Inheritance diagram for std::is_floating_point< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.802.1 Detailed Description

template<typename _Tp>struct std::is_floating_point< _Tp >

is_floating_point

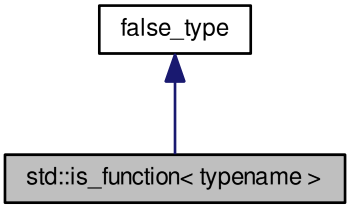
Definition at line 275 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.803 std::is_function< typename > Struct Template Reference

Inheritance diagram for std::is_function< typename >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.803.1 Detailed Description

template<typename>struct std::is_function< typename >

is_function

Definition at line 379 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.804 std::is_fundamental< _Tp > Struct Template Reference

Inherits type< is_arithmetic< _Tp >, is_void< _Tp > >.

4.804.1 Detailed Description

template<typename _Tp>struct std::is_fundamental< _Tp >

is_fundamental

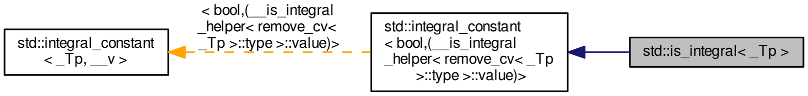
Definition at line 446 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.805 std::is_integral< _Tp > Struct Template Reference

Inheritance diagram for std::is_integral< _Tp >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.805.1 Detailed Description

template<typename _Tp>struct std::is_integral< _Tp >

is_integral

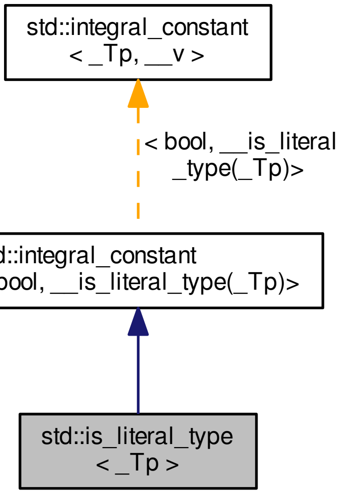
Definition at line 246 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.806 std::is_literal_type< _Tp > Struct Template Reference

Inheritance diagram for std::is_literal_type< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.806.1 Detailed Description

template<typename _Tp>struct std::is_literal_type< _Tp >

is_literal_type

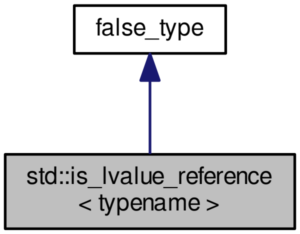
Definition at line 530 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.807 std::is_lvalue_reference< typename > Struct Template Reference

Inheritance diagram for std::is_lvalue_reference< typename >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.807.1 Detailed Description

template<typename>struct std::is_lvalue_reference< typename >

is_lvalue_reference

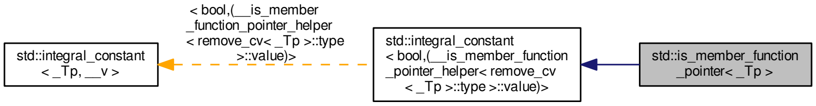
Definition at line 310 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.808 std::is_member_function_pointer< _Tp > Struct Template Reference

Inheritance diagram for std::is_member_function_pointer< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.808.1 Detailed Description

template<typename _Tp>struct std::is_member_function_pointer< _Tp >

is_member_function_pointer

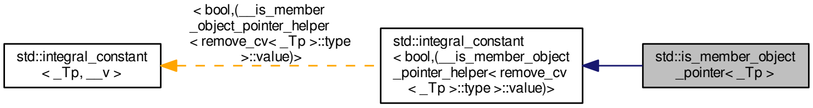
Definition at line 354 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.809 std::is_member_object_pointer< _Tp > Struct Template Reference

Inheritance diagram for std::is_member_object_pointer< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.809.1 Detailed Description

template<typename _Tp>struct std::is_member_object_pointer< _Tp >

is_member_object_pointer

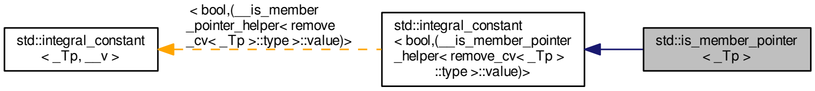
Definition at line 339 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.810 std::is_member_pointer< _Tp > Struct Template Reference

Inheritance diagram for std::is_member_pointer< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.810.1 Detailed Description

template<typename _Tp>struct std::is_member_pointer< _Tp >

is_member_pointer

Definition at line 482 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.811 std::is_move_assignable< _Tp > Struct Template Reference

Inherits __is_move_assignable_impl< _Tp, bool >.

4.811.1 Detailed Description

template<typename _Tp>struct std::is_move_assignable< _Tp >

is_move_assignable

Definition at line 1136 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.812 std::is_move_constructible< _Tp > Struct Template Reference

Inherits __is_move_constructible_impl< _Tp, bool >.

4.812.1 Detailed Description

template<typename _Tp>struct std::is_move_constructible< _Tp >

is_move_constructible

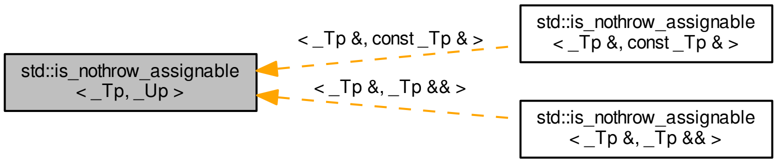
Definition at line 992 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.813 std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference

Inheritance diagram for std::is_nothrow_assignable< _Tp, _Up >:



4.813.1 Detailed Description

template<typename _Tp, typename _Up>struct std::is_nothrow_assignable< _Tp, _Up >

is_nothrow_assignable

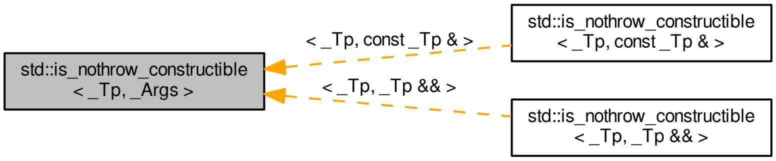
Definition at line 1147 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.814 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference

Inheritance diagram for std::is_nothrow_constructible< _Tp, _Args >:



4.814.1 Detailed Description

template<typename _Tp, typename... _Args>struct std::is_nothrow_constructible< _Tp, _Args >

is_nothrow_constructible

Definition at line 1041 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.815 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference

Inherits __is_nt_copy_assignable_impl< _Tp, bool >.

4.815.1 Detailed Description

template<typename _Tp>struct std::is_nothrow_copy_assignable< _Tp >

is_nothrow_copy_assignable

Definition at line 1166 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.816 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference

Inherits __is_nothrow_copy_constructible_impl< _Tp, bool >.

4.816.1 Detailed Description

template<typename _Tp>struct std::is_nothrow_copy_constructible< _Tp >

is_nothrow_copy_constructible

Definition at line 1060 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.817 std::is_nothrow_default_constructible< _Tp > Struct Template Reference

Inherits type< is_default_constructible< _Tp >, __is_nt_default_constructible_impl< _Tp > >.

Inherited by std::__is_nt_constructible_impl< _Tp >.

4.817.1 Detailed Description

template<typename _Tp>struct std::is_nothrow_default_constructible< _Tp >

is_nothrow_default_constructible

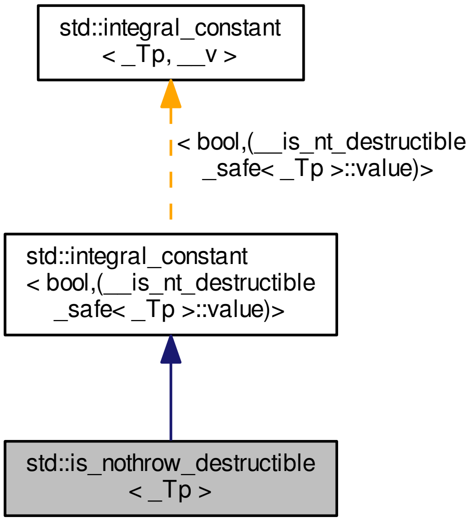
Definition at line 1018 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.818 std::is_nothrow_destructible< _Tp > Struct Template Reference

Inheritance diagram for std::is_nothrow_destructible< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.818.1 Detailed Description

template<typename _Tp>struct std::is_nothrow_destructible< _Tp >

is_nothrow_destructible

Definition at line 700 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.819 std::is_nothrow_move_assignable< _Tp > Struct Template Reference

Inherits __is_nt_move_assignable_impl< _Tp, bool >.

4.819.1 Detailed Description

template<typename _Tp>struct std::is_nothrow_move_assignable< _Tp >

is_nothrow_move_assignable

Definition at line 1184 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.820 std::is_nothrow_move_constructible< _Tp > Struct Template Reference

Inherits __is_nothrow_move_constructible_impl< _Tp, bool >.

4.820.1 Detailed Description

template<typename _Tp>struct std::is_nothrow_move_constructible< _Tp >

is_nothrow_move_constructible

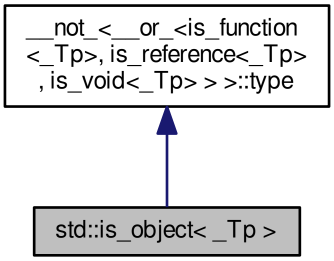
Definition at line 1078 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.821 std::is_object< _Tp > Struct Template Reference

Inheritance diagram for std::is_object< _Tp >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.821.1 Detailed Description

template<typename _Tp>struct std::is_object< _Tp >

is_object

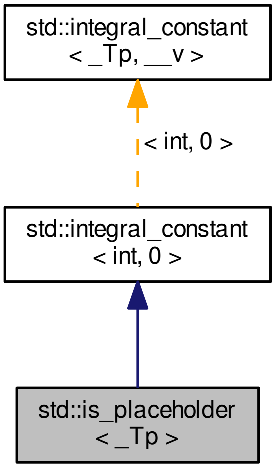
Definition at line 452 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.822 std::is_placeholder< _Tp > Struct Template Reference

Inheritance diagram for std::is_placeholder< _Tp >:



Public Types

- typedef integral_constant< int, __v > **type**
- typedef int **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr int **value**

4.822.1 Detailed Description

template<typename _Tp>struct std::is_placeholder< _Tp >

Determines if the given type _Tp is a placeholder in a bind() expression and, if so, which placeholder it is. [TR1 3.6.2].

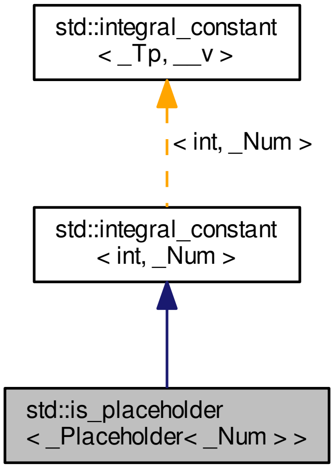
Definition at line 982 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.823 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference

Inheritance diagram for std::is_placeholder< _Placeholder< _Num > >:



Public Types

- typedef integral_constant< int,
  __v > **type**
- typedef int **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr int **value**

4.823.1 Detailed Description

template<int _Num>struct std::is_placeholder< _Placeholder< _Num > >

Partial specialization of is_placeholder that provides the placeholder number for the placeholder objects defined by libstdc++.

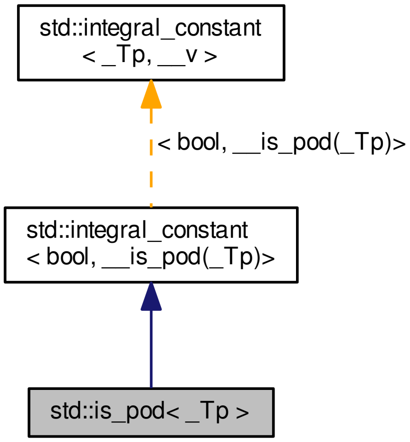
Definition at line 1044 of file functional.

The documentation for this struct was generated from the following file:

- functional

4.824 std::is_pod< _Tp > Struct Template Reference

Inheritance diagram for std::is_pod< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.824.1 Detailed Description

template<typename _Tp>struct std::is_pod< _Tp >

is_pod

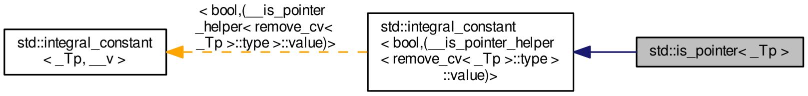
Definition at line 524 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.825 std::is_pointer< _Tp > Struct Template Reference

Inheritance diagram for std::is_pointer< _Tp >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.825.1 Detailed Description

template<typename _Tp>struct std::is_pointer< _Tp >

is_pointer

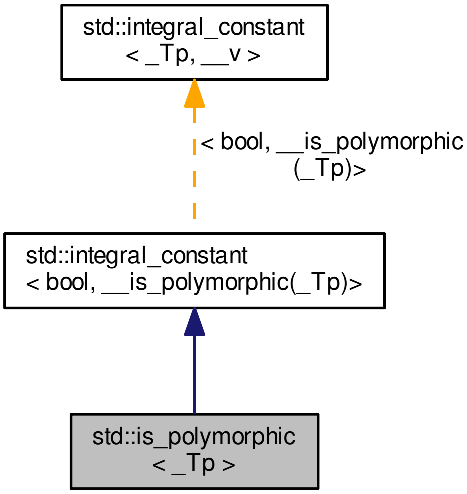
Definition at line 303 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.826 std::is_polymorphic< _Tp > Struct Template Reference

Inheritance diagram for std::is_polymorphic< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.826.1 Detailed Description

template<typename _Tp>struct std::is_polymorphic< _Tp >

is_polymorphic

Definition at line 542 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.827 std::is_reference< _Tp > Struct Template Reference

Inherits type< is_lvalue_reference< _Tp >, is_rvalue_reference< _Tp > >.

4.827.1 Detailed Description

template<typename _Tp>struct std::is_reference< _Tp >

is_reference

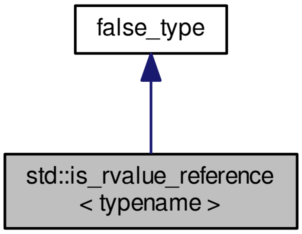
Definition at line 433 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.828 std::is_rvalue_reference< typename > Struct Template Reference

Inheritance diagram for std::is_rvalue_reference< typename >:



Public Types

- typedef integral_constant< _Tp,
  __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.828.1 Detailed Description

template<typename>struct std::is_rvalue_reference< typename >

is_rvalue_reference

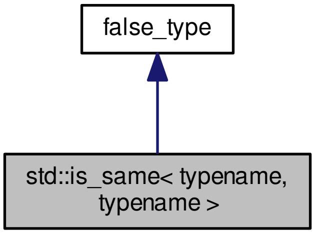
Definition at line 319 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.829 std::is_same< typename, typename > Struct Template Reference

Inheritance diagram for std::is_same< typename, typename >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.829.1 Detailed Description

template<typename, typename>struct std::is_same< typename, typename >

is_same

Definition at line 1278 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.830 std::is_scalar< _Tp > Struct Template Reference

Inherits type< is_arithmetic< _Tp >, is_enum< _Tp >, is_pointer< _Tp >, is_member_pointer< _Tp >, __is_nullptr_t< _Tp > >.

4.830.1 Detailed Description

template<typename _Tp>struct std::is_scalar< _Tp >

is_scalar

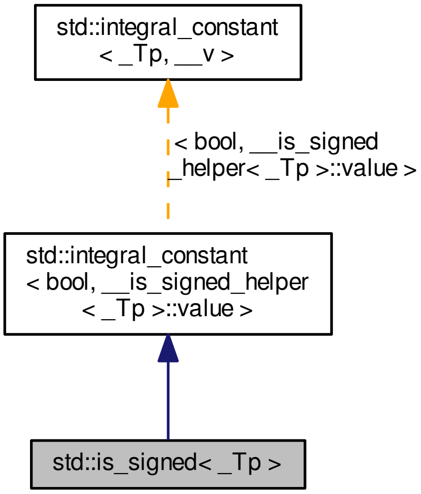
Definition at line 462 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.831 std::is_signed< _Tp > Struct Template Reference

Inheritance diagram for std::is_signed< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.831.1 Detailed Description

template<typename _Tp>struct std::is_signed< _Tp >

is_signed

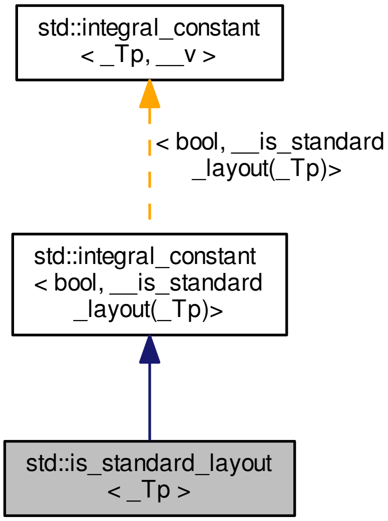
Definition at line 569 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.832 std::is_standard_layout< _Tp > Struct Template Reference

Inheritance diagram for std::is_standard_layout< _Tp >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.832.1 Detailed Description

template<typename _Tp>struct std::is_standard_layout< _Tp >

is_standard_layout

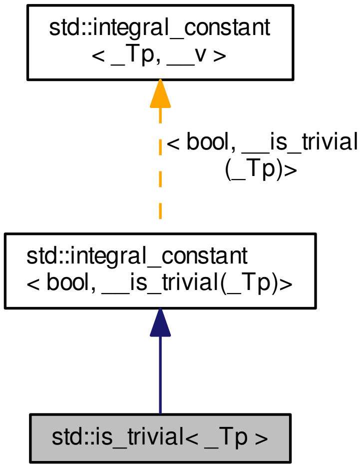
Definition at line 517 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.833 std::is_trivial< _Tp > Struct Template Reference

Inheritance diagram for std::is_trivial< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.833.1 Detailed Description

template<typename _Tp>struct std::is_trivial< _Tp >

is_trivial

Definition at line 509 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.834 std::is_trivially_destructible< _Tp > Struct Template Reference

Inherits type< is_destructible< _Tp >, integral_constant< bool, __has_trivial_destructor(_Tp)> >.

4.834.1 Detailed Description

template<typename _Tp>struct std::is_trivially_destructible< _Tp >

is_trivially_constructible (still unimplemented)

is_trivially_default_constructible (still unimplemented) is_trivially_copy_constructible (still unimplemented) is_trivially_move_constructible (still unimplemented) is_trivially_assignable (still unimplemented) is_trivially_copy_assignable (still unimplemented) is_trivially_move_assignable (still unimplemented) is_trivially_destructible

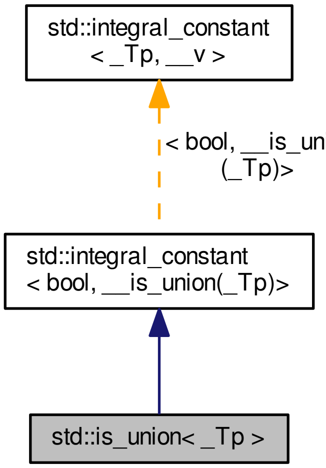
Definition at line 1204 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.835 std::is_union< _Tp > Struct Template Reference

Inheritance diagram for std::is_union< _Tp >:



Public Types

- typedef integral_constant
< bool, __v > **type**

- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.835.1 Detailed Description

template<typename _Tp>struct std::is_union< _Tp >

is_union

Definition at line 367 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.836 std::is_unsigned< _Tp > Struct Template Reference

Inherits type< is_arithmetic< _Tp >, __not_< is_signed< _Tp > > >.

4.836.1 Detailed Description

template<typename _Tp>struct std::is_unsigned< _Tp >

is_unsigned

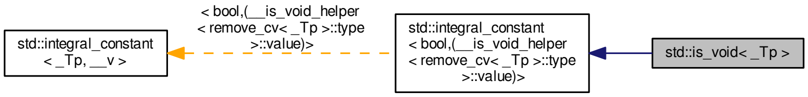
Definition at line 575 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.837 std::is_void< _Tp > Struct Template Reference

Inheritance diagram for std::is_void< _Tp >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.837.1 Detailed Description

template<typename _Tp>struct std::is_void< _Tp >

is_void

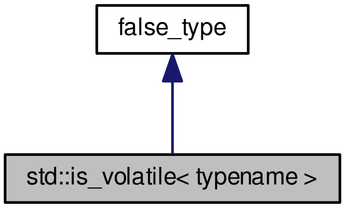
Definition at line 163 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.838 std::is_volatile< typename > Struct Template Reference

Inheritance diagram for std::is_volatile< typename >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.838.1 Detailed Description

template<typename>struct std::is_volatile< typename >

is_volatile

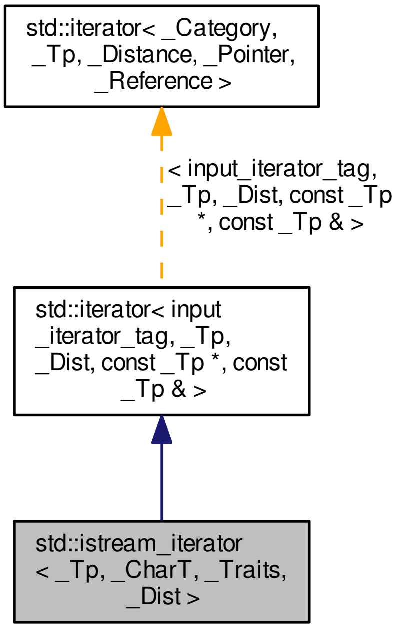
Definition at line 500 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.839 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference

Inheritance diagram for std::istream_iterator< _Tp, _CharT, _Traits, _Dist >:



Public Types

- typedef _CharT **char_type**
- typedef _Dist difference_type
- typedef basic_istream< _CharT, _Traits > **istream_type**
- typedef input_iterator_tag iterator_category
- typedef const _Tp ∗ pointer
- typedef const _Tp & reference
- typedef _Traits **traits_type**
- typedef _Tp value_type

Public Member Functions

- constexpr istream_iterator ()
- istream_iterator (istream_type &__s)
- **istream_iterator** (const istream_iterator &__obj)
- bool **_M_equal** (const istream_iterator &__x) const
- const _Tp & **operator∗** () const
- istream_iterator & **operator++** ()
- istream_iterator **operator++** (int)
- const _Tp ∗ **operator->** () const

4.839.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >

Provides input iterator semantics for streams.

Definition at line 49 of file stream_iterator.h.

4.839.2 Member Typedef Documentation

4.839.2.1 typedef _Dist std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp ∗ , const _Tp & >::difference_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.839.2.2 typedef input_iterator_tag std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp ∗ , const _Tp & >::iterator_category [inherited]

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.839.2.3 typedef const _Tp ∗ std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp ∗ , const _Tp & >::pointer [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.839.2.4 typedef const _Tp & **std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::reference** `[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.839.2.5 typedef _Tp **std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type** `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.839.3 Constructor & Destructor Documentation

4.839.3.1 template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> constexpr **std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ( )** `[inline]`

Construct end of input stream iterator.

Definition at line 64 of file stream_iterator.h.

4.839.3.2 template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t> **std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (istream_type & *__s*)** `[inline]`

Construct start of input stream iterator.

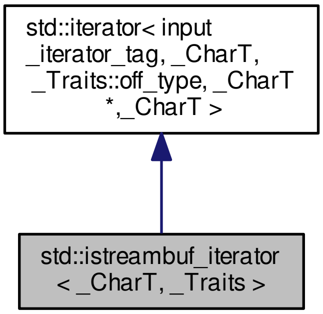
Definition at line 68 of file stream_iterator.h.

The documentation for this class was generated from the following file:

- stream_iterator.h

4.840 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::istreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef _Distance difference_type
- typedef _Category iterator_category
- typedef _Pointer pointer
- typedef _Reference reference
- typedef _Tp value_type

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef _Traits::int_type int_type
- typedef basic_streambuf
  < _CharT, _Traits > streambuf_type
- typedef basic_istream< _CharT,
  _Traits > istream_type

Public Member Functions

- constexpr istreambuf_iterator () noexcept
- **istreambuf_iterator** (const istreambuf_iterator &) noexcept=default
- istreambuf_iterator (istream_type &__s) noexcept
- istreambuf_iterator (streambuf_type ∗__s) noexcept
- bool equal (const istreambuf_iterator &__b) const
- char_type operator∗ () const
- istreambuf_iterator & operator++ ()
- istreambuf_iterator operator++ (int)

Friends

- template< bool _IsMove, typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, _CharT2 ∗ >::__type **__copy_move_a2** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 ∗)
- template< typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, ostreambuf_iterator
  < _CharT2 > >::__type **copy** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)
- template< typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, istreambuf_iterator
  < _CharT2 > >::__type **find** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)

4.840.1 Detailed Description

template<typename _CharT, typename _Traits>class std::istreambuf_iterator< _CharT, _Traits >

Provides input iterator semantics for streambufs.

Definition at line 50 of file streambuf_iterator.h.

4.840.2 Member Typedef Documentation

4.840.2.1 template<typename _CharT , typename _Traits > typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type

Public typedefs.

Definition at line 64 of file streambuf_iterator.h.

4.840.2.2 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.840.2.3 template<typename _CharT , typename _Traits > typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type

Public typedefs.

Definition at line 66 of file streambuf_iterator.h.

4.840.2.4 template<typename _CharT , typename _Traits > typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type

Public typedefs.

Definition at line 68 of file streambuf_iterator.h.

4.840.2.5 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category [inherited]

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.840.2.6 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.840.2.7 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Reference **std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference** [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.840.2.8 template<typename _CharT , typename _Traits > typedef **basic_streambuf**<_CharT, _Traits> **std::istreambuf_iterator< _CharT, _Traits >::streambuf_type**

Public typedefs.

Definition at line 67 of file streambuf_iterator.h.

4.840.2.9 template<typename _CharT , typename _Traits > typedef _Traits **std::istreambuf_iterator< _CharT, _Traits >::traits_type**

Public typedefs.

Definition at line 65 of file streambuf_iterator.h.

4.840.2.10 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&> typedef _Tp **std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type** [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.840.3 Constructor & Destructor Documentation

4.840.3.1 template<typename _CharT , typename _Traits > constexpr **std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator ( )** [inline], [noexcept]

Construct end of input stream iterator.

Definition at line 102 of file streambuf_iterator.h.

4.840.3.2 template<typename _CharT , typename _Traits > **std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (istream_type & *__s*)** [inline], [noexcept]

Construct start of input stream iterator.

Definition at line 112 of file streambuf_iterator.h.

4.840.3.3 template<typename _CharT , typename _Traits > **std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (streambuf_type * *__s*)** [inline], [noexcept]

Construct start of streambuf iterator.

Definition at line 116 of file streambuf_iterator.h.

4.840.4 Member Function Documentation

4.840.4.1 template<typename _CharT , typename _Traits > bool std::istreambuf_iterator< _CharT, _Traits >::equal (const istreambuf_iterator< _CharT, _Traits > & *__b*) const [inline]

Return true both iterators are end or both are not end.

Definition at line 172 of file streambuf_iterator.h.

4.840.4.2 template<typename _CharT , typename _Traits > char_type std::istreambuf_iterator< _CharT, _Traits >::operator* ( ) const [inline]

Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator*() on an end of stream is undefined.

Definition at line 123 of file streambuf_iterator.h.

4.840.4.3 template<typename _CharT , typename _Traits > istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits >::operator++ ( ) [inline]

Advance the iterator. Calls streambuf.sbumpc().

Definition at line 137 of file streambuf_iterator.h.

References std::basic_streambuf< _CharT, _Traits >::sbumpc().

4.840.4.4 template<typename _CharT , typename _Traits > istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (int) [inline]

Advance the iterator. Calls streambuf.sbumpc().

Definition at line 152 of file streambuf_iterator.h.

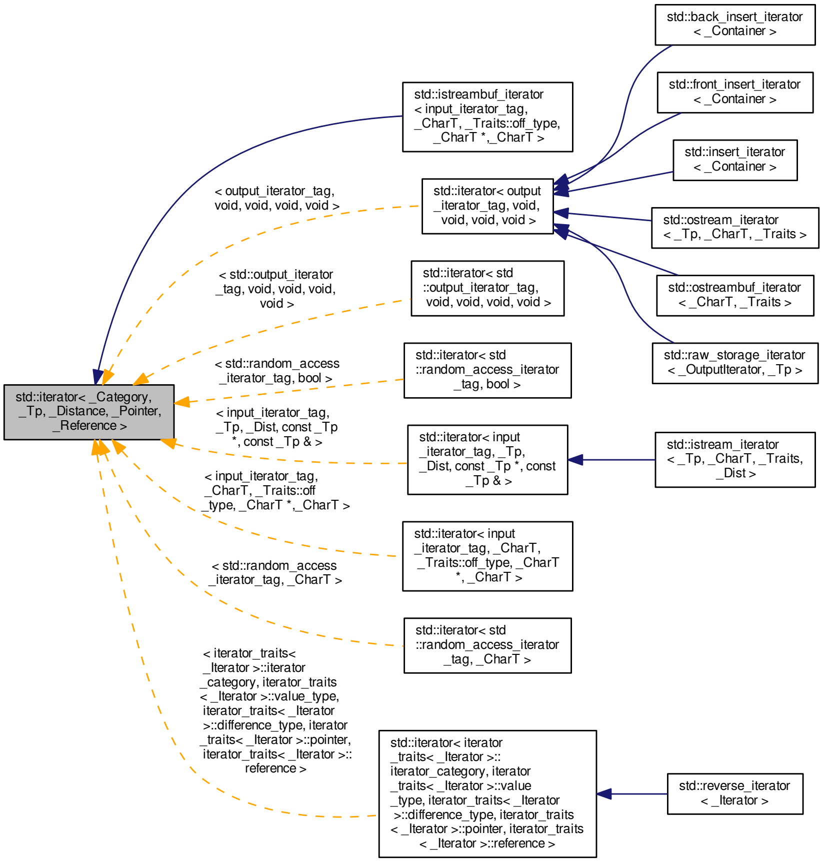
References std::basic_streambuf< _CharT, _Traits >::sbumpc().

The documentation for this class was generated from the following file:

- streambuf_iterator.h

4.841 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference

Inheritance diagram for std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >:



Public Types

- typedef _Distance difference_type
- typedef _Category iterator_category
- typedef _Pointer pointer
- typedef _Reference reference
- typedef _Tp value_type

4.841.1 Detailed Description

template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp∗, typename _Reference = _Tp&>struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 118 of file stl_iterator_base_types.h.

4.841.2 Member Typedef Documentation

4.841.2.1 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp∗, typename _Reference = _Tp&> typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.841.2.2 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp∗, typename _Reference = _Tp&> typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.841.2.3 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp∗, typename _Reference = _Tp&> typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.841.2.4 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp∗, typename _Reference = _Tp&> typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.841.2.5 template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp∗, typename _Reference = _Tp&> typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.842 std::iterator_traits< _Tp ∗ > Struct Template Reference

Public Types

- typedef ptrdiff_t **difference_type**
- typedef random_access_iterator_tag **iterator_category**
- typedef _Tp ∗ **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

4.842.1 Detailed Description

template<typename _Tp>struct std::iterator_traits< _Tp * >

Partial specialization for pointer types.

Definition at line 175 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.843 std::iterator_traits< const _Tp * > Struct Template Reference

Public Types

- typedef ptrdiff_t **difference_type**
- typedef random_access_iterator_tag **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

4.843.1 Detailed Description

template<typename _Tp>struct std::iterator_traits< const _Tp * >

Partial specialization for const pointer types.

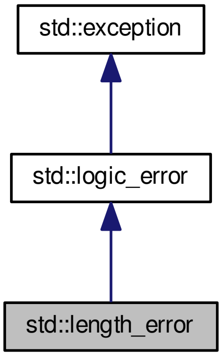
Definition at line 186 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.844 std::length_error Class Reference

Inheritance diagram for std::length_error:



Public Member Functions

- **length_error** (const string &__arg)
- virtual const char ∗ what () const noexcept

4.844.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a basic_string instance).

Definition at line 91 of file stdexcept.

4.844.2 Member Function Documentation

4.844.2.1 virtual const char∗ std::logic_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

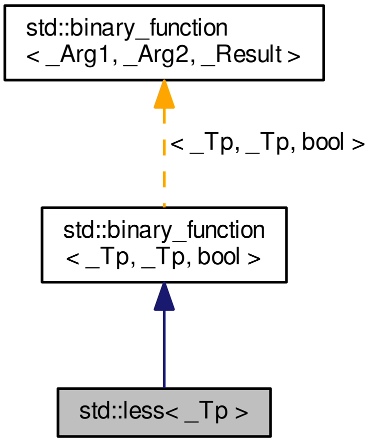
Reimplemented in std::future_error.

The documentation for this class was generated from the following file:

- stdexcept

4.845 std::less< _Tp > Struct Template Reference

Inheritance diagram for std::less< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.845.1 Detailed Description

template<typename _Tp>struct std::less< _Tp >

One of the comparison functors.

Definition at line 231 of file stl_function.h.

4.845.2 Member Typedef Documentation

4.845.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.845.2.2 typedef bool **std::binary_function< _Tp , _Tp , bool >::result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.845.2.3 typedef _Tp **std::binary_function< _Tp , _Tp , bool >::second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

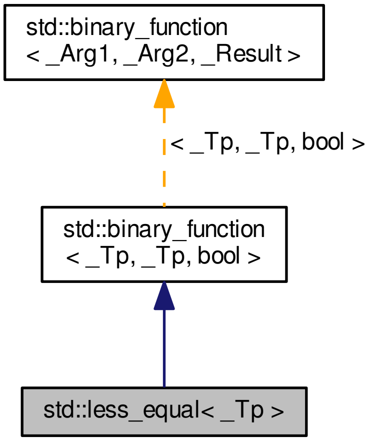
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.846 std::less_equal< _Tp > Struct Template Reference

Inheritance diagram for std::less_equal< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.846.1 Detailed Description

template<typename _Tp>struct std::less_equal< _Tp >

One of the comparison functors.

Definition at line 249 of file stl_function.h.

4.846.2 Member Typedef Documentation

4.846.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.846.2.2 typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.846.2.3 typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.847 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference

Public Types

- typedef _UIntType result_type

Public Member Functions

- linear_congruential_engine (result_type __s=default_seed)
- template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value> ::type>
  linear_congruential_engine (_Sseq &__q)
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed (result_type __s=default_seed)
- template<typename _Sseq >
  std::enable_if< std::is_class
  < _Sseq >::value >::type seed (_Sseq &__q)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr result_type increment
- static constexpr result_type modulus
- static constexpr result_type multiplier

Friends

- template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT , typename _Traits > std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &__lcr)
- bool operator== (const linear_congruential_engine &__lhs, const linear_congruential_engine &__rhs)
- template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT , typename _Traits > std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &__lcr)

4.847.1 Detailed Description

template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>class std::linear_congruential_engine< _UIntType, __a, __c, __m >

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to (__m-1). If the template parameter `__m` is 0, the modulus `__m` used is std::numeric_limits<_UIntType>::max() plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 241 of file random.h.

4.847.2 Member Typedef Documentation

4.847.2.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type

The type of the generated random value.

Definition at line 244 of file random.h.

4.847.3 Constructor & Destructor Documentation

4.847.3.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (result_type *__s* = `default_seed`) `[inline]`, `[explicit]`

Constructs a linear_congruential_engine random number generator engine with seed `__s`. The default seed value is 1.

Parameters

__s	The initial seed value.

Definition at line 268 of file random.h.

References std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

4.847.3.2 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, linear_congruential_engine>::value> ::type> std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (_Sseq & *__q*) [inline], [explicit]

Constructs a linear_congruential_engine random number generator engine seeded from the seed sequence `__q`.

Parameters

__q	the seed sequence.

Definition at line 281 of file random.h.

References std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

4.847.4 Member Function Documentation

4.847.4.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> void std::linear_congruential_engine< _UIntType, __a, __c, __m >::discard (unsigned long long *__z*) [inline]

Discard a sequence of random numbers.

Definition at line 325 of file random.h.

4.847.4.2 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::max ( ) [inline], [static]

Gets the largest possible value in the output range.

Definition at line 318 of file random.h.

4.847.4.3 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::min ( ) [inline], [static]

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be > 0, otherwise 0 is allowed.

Definition at line 311 of file random.h.

4.847.4.4 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator() ( ) [inline]

Gets the next random number in the sequence.

Definition at line 335 of file random.h.

4.847.4.5 template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m> void **std::linear_congruential_engine**< _UIntType, __a, __c, __m >::seed (**result_type** ***__s*** = `default_seed`)

Reseeds the linear_congruential_engine random number generator engine sequence to the seed `__s`.

Parameters

__s	The new seed.

Seeds the LCR with integral value `__s`, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 120 of file bits/random.tcc.

Referenced by std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine().

4.847.4.6 template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m> template<typename _Sseq > **std::enable_if**< **std::is_class**< _Sseq >::value >::type **std::linear_congruential_engine**< _UIntType, __a, __c, __m >::seed (_Sseq & ***__q***)

Reseeds the linear_congruential_engine random number generator engine sequence using values from the seed sequence `__q`.

Parameters

__q	the seed sequence.

Seeds the LCR engine with a value generated by `__q`.

Definition at line 136 of file bits/random.tcc.

References std::__lg().

4.847.5 Friends And Related Function Documentation

4.847.5.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT , typename _Traits > **std::basic_ostream**<_CharT, _Traits>& operator<< (**std::basic_ostream**< _CharT, _Traits > & ***__os***, const **std::linear_congruential_engine**< _UIntType1, __a1, __c1, __m1 > & ***__lcr***) `[friend]`

Writes the textual representation of the state x(i) of x to `__os`.

Parameters

__os	The output stream.
__lcr	A % linear_congruential_engine random number generator.

Returns

__os.

4.847.5.2 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> bool operator== (const **linear_congruential_engine**< _UIntType, __a, __c, __m > & ***__lhs***, const **linear_congruential_engine**< _UIntType, __a, __c, __m > & ***__rhs***) `[friend]`

Compares two linear congruential random number generator objects of the same type for equality.

Parameters

__lhs	A linear congruential random number generator object.
__rhs	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 353 of file random.h.

4.847.5.3 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> template<typename _UIntType1 , _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & *__is,* std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & *__lcr*) `[friend]`

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments _CharT and _Traits were the same as those of `__is`.

Parameters

__is	The input stream.
__lcr	A % linear_congruential_engine random number generator.

Returns

__is.

4.847.6 Member Data Documentation

4.847.6.1 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment `[static]`

An increment.

Definition at line 255 of file random.h.

4.847.6.2 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus `[static]`

The modulus.

Definition at line 257 of file random.h.

4.847.6.3 template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m> constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier `[static]`

The multiplier.

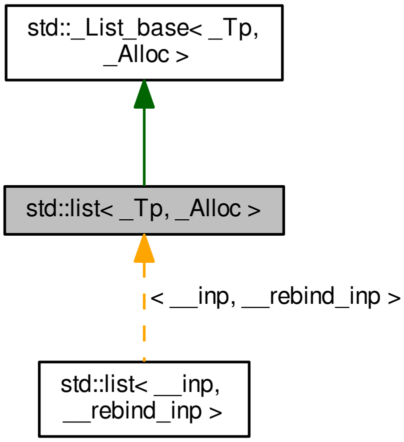
Definition at line 253 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.848 std::list< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _List_const_iterator< _Tp > **const_iterator**
- typedef _Tp_alloc_type::const_pointer **const_pointer**
- typedef _Tp_alloc_type::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _List_iterator< _Tp > **iterator**
- typedef _Tp_alloc_type::pointer **pointer**
- typedef _Tp_alloc_type::reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- list ()
- list (const allocator_type &__a)
- list (size_type __n)
- list (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())

- list (const list &__x)
- list (list &&__x) noexcept
- list (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- void assign (size_type __n, const value_type &__val)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- reference back ()
- const_reference back () const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (iterator __position, _Args &&...__args)
- template<typename... _Args>
  void **emplace_back** (_Args &&...__args)
- template<typename... _Args>
  void **emplace_front** (_Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- iterator insert (iterator __position, const value_type &__x)
- iterator insert (iterator __position, value_type &&__x)
- void insert (iterator __p, initializer_list< value_type > __l)
- void insert (iterator __position, size_type __n, const value_type &__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void insert (iterator __position, _InputIterator __first, _InputIterator __last)
- size_type max_size () const noexcept
- void merge (list &&__x)
- void **merge** (list &__x)
- template<typename _StrictWeakOrdering >
  void merge (list &&__x, _StrictWeakOrdering __comp)
- template<typename _StrictWeakOrdering >
  void **merge** (list &__x, _StrictWeakOrdering __comp)
- list & operator= (const list &__x)
- list & operator= (list &&__x)
- list & operator= (initializer_list< value_type > __l)
- void pop_back ()
- void pop_front ()
- void push_back (const value_type &__x)

- void **push_back** (value_type &&__x)
- void push_front (const value_type &__x)
- void **push_front** (value_type &&__x)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- void remove (const _Tp &__value)
- template<typename _Predicate >
  void remove_if (_Predicate)
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void reverse () noexcept
- size_type size () const noexcept
- void sort ()
- template<typename _StrictWeakOrdering >
  void sort (_StrictWeakOrdering)
- void splice (iterator __position, list &&__x)
- void **splice** (iterator __position, list &__x)
- void splice (iterator __position, list &&__x, iterator __i)
- void **splice** (iterator __position, list &__x, iterator __i)
- void splice (iterator __position, list &&__x, iterator __first, iterator __last)
- void **splice** (iterator __position, list &__x, iterator __first, iterator __last)
- void swap (list &__x)
- void unique ()
- template<typename _BinaryPredicate >
  void unique (_BinaryPredicate)

Protected Types

- typedef _List_node< _Tp > **_Node**

Protected Member Functions

- template<typename _Integer >
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator >
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_check_equal_allocators** (list &__x)
- void **_M_clear** ()
- template<typename... _Args>
  _Node ∗ _M_create_node (_Args &&...__args)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_erase** (iterator __position)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__x)
- _List_node< _Tp > ∗ **_M_get_node** ()
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept

- _Tp_alloc_type **_M_get_Tp_allocator** () const noexcept
- void **_M_init** ()
- template<typename _Integer >
  void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
  void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename... _Args>
  void **_M_insert** (iterator __position, _Args &&...__args)
- void **_M_put_node** (_List_node< _Tp > ∗__p)
- void **_M_transfer** (iterator __position, iterator __first, iterator __last)

Protected Attributes

- _List_impl **_M_impl**

4.848.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>class std::list< _Tp, _Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator`[].

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike std::vector and std::deque, random-access iterators are not provided, so subscripting ([]) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, std::list provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for list<Tp>:

First, we never actually allocate a Tp, we allocate List_node<Tp>'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from list<X,Alloc1> are spliced into list<X,Alloc2>, destroying the memory of the second list is a valid operation, i.e., Alloc1 giveth and Alloc2 taketh away.

Second, a list conceptually represented as

```
A <---> B <---> C <---> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private list::iterator pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's next/previous pointers refer to itself, the list is empty.

Definition at line 438 of file stl_list.h.

4.848.2 Constructor & Destructor Documentation

4.848.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list ( ) [inline]

Default constructor creates no elements.

Definition at line 523 of file stl_list.h.

4.848.2.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list (const allocator_type & *__a*) [inline], [explicit]

Creates a list with no elements.

Parameters

__a	An allocator object.

Definition at line 531 of file stl_list.h.

4.848.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list (size_type *__n*) [inline], [explicit]

Creates a list with default constructed elements.

Parameters

__n	The number of elements to initially create.

This constructor fills the list with *__n* default constructed elements.

Definition at line 543 of file stl_list.h.

4.848.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list (size_type *__n*, const value_type & *__value*, const allocator_type & *__a* = allocator_type()) [inline]

Creates a list with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator object.

This constructor fills the list with *__n* copies of *__value*.

Definition at line 555 of file stl_list.h.

4.848.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list (const list< _Tp, _Alloc > & *__x*) [inline]

List copy constructor.

Parameters

__x	A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *__x*.

Definition at line 582 of file stl_list.h.

4.848.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list (list< _Tp, _Alloc > && __x) [inline], [noexcept]

List move constructor.

Parameters

__x	A list of identical element and allocator types.

The newly-created list contains the exact contents of *__x*. The contents of *__x* are a valid, but unspecified list.

Definition at line 594 of file stl_list.h.

4.848.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::list< _Tp, _Alloc >::list (initializer_list< value_type > __l, const allocator_type & __a = allocator_type()) [inline]

Builds a list from an initializer_list.

Parameters

__l	An initializer_list of value_type.
__a	An allocator object.

Create a list consisting of copies of the elements in the initializer_list *__l*. This is linear in __l.size().

Definition at line 605 of file stl_list.h.

4.848.2.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> std::list< _Tp, _Alloc >::list (_InputIterator __first, _InputIterator __last, const allocator_type & __a = allocator_type()) [inline]

Builds a list from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__a	An allocator object.

Create a list consisting of copies of the elements from [*__first*,*__last*). This is linear in N (where N is distance(*__first*,*__last*)).

Definition at line 624 of file stl_list.h.

4.848.3 Member Function Documentation

4.848.3.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename... _Args> _Node* std::list< _Tp, _Alloc >::_M_create_node (_Args &&... __args) [inline], [protected]

Parameters

__args	An instance of user data.

Allocates space for a new node and constructs a copy of *__args* in it.

Definition at line 500 of file stl_list.h.

4.848.3.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void **std::list**< _Tp, _Alloc >::assign (size_type *__n*, const value_type & *__val*) [inline]

Assigns a given value to a list.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a list with *__n* copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 702 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::operator=().

4.848.3.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void **std::list**< _Tp, _Alloc >::assign (_InputIterator *__first*, _InputIterator *__last*) [inline]

Assigns a range to a list.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a list with copies of the elements in the range [*__first*,*__last*).

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 721 of file stl_list.h.

4.848.3.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void **std::list**< _Tp, _Alloc >::assign (**initializer_list**< value_type > *__l*) [inline]

Assigns an initializer_list to a list.

Parameters

__l	An initializer_list of value_type.

Replace the contents of the list with copies of the elements in the initializer_list *__l*. This is linear in __l.size().

Definition at line 743 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::assign().

4.848.3.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference **std::list**< _Tp, _Alloc >::back ( ) [inline]

Returns a read/write reference to the data at the last element of the list.

Definition at line 943 of file stl_list.h.

4.848.3.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list< _Tp, _Alloc >::back ( ) const [inline]

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 955 of file stl_list.h.

4.848.3.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 758 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::crend(), std::list< __inp, __rebind_inp >::front(), std::list< __inp, __rebind_inp >::list(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::operator=(), std::operator==(), std::list< __inp, __rebind_inp >::pop_front(), std::list< __inp, __rebind_inp >::push_front(), std::list< __inp, __rebind_inp >::rend(), std::list< __inp, __rebind_inp >::size(), std::list< _Tp, _Alloc >::sort(), and std::list< __inp, __rebind_inp >::splice().

4.848.3.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 767 of file stl_list.h.

4.848.3.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 831 of file stl_list.h.

4.848.3.10 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 840 of file stl_list.h.

4.848.3.11 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::clear ( ) [inline], [noexcept]

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1228 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::operator=().

4.848.3.12 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list< _Tp, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 849 of file stl_list.h.

4.848.3.13 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 858 of file stl_list.h.

4.848.3.14 template<typename _Tp , typename _Alloc > template<typename... _Args> list< _Tp, _Alloc >::iterator list::emplace (iterator __position, _Args &&... __args)

Constructs object in list before specified iterator.

Parameters

__position	A const_iterator into the list.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 88 of file list.tcc.

Referenced by std::list< __inp, __rebind_inp >::insert().

4.848.3.15 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::list< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the list is empty. (Thus begin() would equal end().)

Definition at line 868 of file stl_list.h.

Referenced by std::list< _Tp, _Alloc >::sort(), and std::list< __inp, __rebind_inp >::splice().

4.848.3.16 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 776 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::back(), std::list< __inp, __rebind_inp >::crbegin(), std::list< __inp, __rebind_inp >::list(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::operator=(), std::operator==(), std::list< __inp, __rebind_inp >::push_back(), std::list< __inp, __rebind_inp >::rbegin(), std::list< __inp, __rebind_inp >::size(), and std::list< __inp, __rebind_inp >::splice().

4.848.3.17 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::list< _Tp, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 785 of file stl_list.h.

4.848.3.18 template<typename _Tp , typename _Alloc > list< _Tp, _Alloc >::iterator list::erase (iterator __position)

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 109 of file list.tcc.

Referenced by std::list< __inp, __rebind_inp >::erase().

4.848.3.19 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::erase (iterator *__first,* iterator *__last*) [inline]

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1193 of file stl_list.h.

4.848.3.20 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::list< _Tp, _Alloc >::front ( ) [inline]

Returns a read/write reference to the data at the first element of the list.

Definition at line 927 of file stl_list.h.

4.848.3.21 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::list< _Tp, _Alloc >::front ( ) const [inline]

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 935 of file stl_list.h.

4.848.3.22 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> allocator_type std::list< _Tp, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Get a copy of the memory allocation object.

Definition at line 749 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::insert().

4.848.3.23 template<typename _Tp , typename _Alloc > list< _Tp, _Alloc >::iterator list::insert (iterator *__position,* const value_type & *__x*)

Inserts given value into list before specified iterator.

Parameters

__position	An iterator into the list.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 99 of file list.tcc.

4.848.3.24 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::list< _Tp, _Alloc >::insert (iterator *__position,* value_type && *__x*) [inline]

Inserts given rvalue into list before specified iterator.

Parameters

__position	An iterator into the list.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1089 of file stl_list.h.

4.848.3.25 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::insert (iterator *__p,* initializer_list< value_type > *__l*) [inline]

Inserts the contents of an initializer_list into list before specified iterator.

Parameters

__p	An iterator into the list.
__l	An initializer_list of value_type.

This function will insert copies of the data in the initializer_list *l* into the list before the location specified by *p*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1106 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::insert().

4.848.3.26 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::insert (iterator *__position,* size_type *__n,* const value_type & *__x*) [inline]

Inserts a number of copies of given data into the list.

Parameters

__position	An iterator into the list.
__n	Number of elements to be inserted.
__x	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1123 of file stl_list.h.

4.848.3.27 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void std::list< _Tp, _Alloc >::insert (iterator *__position,* _InputIterator *__first,* _InputIterator *__last*) [inline]

Inserts a range into the list.

Parameters

__position	An iterator into the list.
__first	An input iterator.
__last	An input iterator.

This function will insert copies of the data in the range [*first*,*last*) into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1149 of file stl_list.h.

4.848.3.28 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the size() of the largest possible list.

Definition at line 878 of file stl_list.h.

4.848.3.29 template<typename _Tp , typename _Alloc > void list::merge (list< _Tp, _Alloc > && *__x*)

Merge sorted lists.

Parameters

__x	Sorted list to merge.

Assumes that both *__x* and this list are sorted according to operator<(). Merges elements of *__x* into this list in sorted order, leaving *__x* empty when complete. Elements in this list precede elements in *__x* that are equal.

Definition at line 288 of file list.tcc.

References std::begin(), std::list< _Tp, _Alloc >::begin(), std::end(), and std::list< _Tp, _Alloc >::end().

Referenced by std::list< _Tp, _Alloc >::sort().

4.848.3.30 template<typename _Tp , typename _Alloc > template<typename _StrictWeakOrdering > void list::merge (list< _Tp, _Alloc > && *__x*, _StrictWeakOrdering *__comp*)

Merge sorted lists according to comparison function.

Template Parameters

_StrictWeakOrdering	Comparison function defining sort order.

Parameters

__x	Sorted list to merge.
__comp	Comparison functor.

Assumes that both *__x* and this list are sorted according to StrictWeakOrdering. Merges elements of *__x* into this list in sorted order, leaving *__x* empty when complete. Elements in this list precede elements in *__x* that are equivalent according to StrictWeakOrdering().

Definition at line 322 of file list.tcc.

References std::begin(), std::list< _Tp, _Alloc >::begin(), std::end(), and std::list< _Tp, _Alloc >::end().

4.848.3.31 template<typename _Tp , typename _Alloc > list< _Tp, _Alloc > & list::operator= (const list< _Tp, _Alloc > & *__x*)

List assignment operator.

No explicit dtor needed as the _Base dtor takes care of things. The _Base dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

__x	A list of identical element and allocator types.

All the elements of *__x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 185 of file list.tcc.

References std::begin(), std::list< _Tp, _Alloc >::begin(), std::end(), and std::list< _Tp, _Alloc >::end().

4.848.3.32 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list< _Tp, _Alloc >::operator= (list< _Tp, _Alloc > && *__x*) `[inline]`

List move assignment operator.

Parameters

__x	A list of identical element and allocator types.

The contents of *__x* are moved into this list (without copying). *__x* is a valid, but unspecified list

Definition at line 667 of file stl_list.h.

4.848.3.33 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> list& std::list< _Tp, _Alloc >::operator= (initializer_list< value_type > *__l*) `[inline]`

List initializer list assignment operator.

Parameters

__l	An initializer_list of value_type.

Replace the contents of the list with copies of the elements in the initializer_list *__l*. This is linear in l.size().

Definition at line 684 of file stl_list.h.

4.848.3.34 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::pop_back ( ) [inline]

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

Definition at line 1041 of file stl_list.h.

4.848.3.35 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::pop_front ( ) [inline]

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

Definition at line 1001 of file stl_list.h.

4.848.3.36 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::push_back (const value_type & *__x*) [inline]

Add data to the end of the list.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1015 of file stl_list.h.

4.848.3.37 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::push_front (const value_type & *__x*) [inline]

Add data to the front of the list.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 974 of file stl_list.h.

4.848.3.38 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list< _Tp, _Alloc >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 794 of file stl_list.h.

4.848.3.39 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 803 of file stl_list.h.

4.848.3.40 template<typename _Tp, typename _Alloc > void list::remove (const _Tp & *__value*)

Remove all elements equal to value.

Parameters

__value	The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 239 of file list.tcc.

References std::__addressof(), std::begin(), and std::end().

4.848.3.41 template<typename _Tp , typename _Alloc > template<typename _Predicate > void list::remove_if (_Predicate *__pred*)

Remove all elements satisfying a predicate.

Template Parameters

_Predicate	Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 392 of file list.tcc.

References std::begin(), and std::end().

4.848.3.42 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::list< _Tp, _Alloc >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 812 of file stl_list.h.

4.848.3.43 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::list< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 821 of file stl_list.h.

4.848.3.44 template<typename _Tp , typename _Alloc > void list::resize (size_type *__new_size*)

Resizes the list to the specified number of elements.

Parameters

__new_size	Number of elements the list should contain.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 139 of file list.tcc.

References std::begin(), and std::end().

4.848.3.45 template<typename _Tp , typename _Alloc > void list::resize (size_type *__new_size,* const value_type & *__x*)

Resizes the list to the specified number of elements.

Parameters

__new_size	Number of elements the list should contain.
__x	Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 154 of file list.tcc.

References std::begin(), and std::end().

4.848.3.46 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::reverse ( ) [inline], [noexcept]

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1449 of file stl_list.h.

4.848.3.47 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::list< _Tp, _Alloc >::size ( ) const [inline], [noexcept]

Returns the number of elements in the list.

Definition at line 873 of file stl_list.h.

4.848.3.48 template<typename _Tp , typename _Alloc > void list::sort ( )

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 354 of file list.tcc.

References std::begin(), std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::empty(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::splice(), and std::list< _Tp, _Alloc >::swap().

4.848.3.49 template<typename _Tp , typename _Alloc > template<typename _StrictWeakOrdering > void list::sort (_StrictWeakOrdering *__comp*)

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 431 of file list.tcc.

References std::begin(), std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::empty(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::splice(), and std::list< _Tp, _Alloc >::swap().

4.848.3.50 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice (iterator *__position,* list< _Tp, _Alloc > && *__x*) [inline]

Insert contents of another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.

The elements of *__x* are inserted in constant time in front of the element referenced by *__position*. *__x* becomes an empty list.

Requires this != *__x*.

Definition at line 1248 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::insert(), and std::list< _Tp, _Alloc >::sort().

4.848.3.51 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice (iterator *__position,* list< _Tp, _Alloc > && *__x,* iterator *__i*) [inline]

Insert element from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__i	Iterator referencing the element to move.

Removes the element in list *__x* referenced by *__i* and inserts it into the current list before *__position*.

Definition at line 1278 of file stl_list.h.

4.848.3.52 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::splice (iterator *__position,* list< _Tp, _Alloc > && *__x,* iterator *__first,* iterator *__last*) [inline]

Insert range from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__first	Iterator referencing the start of range in x.
__last	Iterator referencing the end of range in x.

Removes elements in the range [__first,__last) and inserts them before *__position* in constant time.

Undefined if __*position* is in [__first,__last).

Definition at line 1314 of file stl_list.h.

4.848.3.53 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::list< _Tp, _Alloc >::swap (list< _Tp, _Alloc > & __*x*) [inline]

Swaps data with another list.

Parameters

__*x*	A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global std::swap() function is specialized such that std::swap(l1,l2) will feed to this function.

Definition at line 1210 of file stl_list.h.

Referenced by std::list< __inp, __rebind_inp >::operator=(), std::list< _Tp, _Alloc >::sort(), and std::swap().

4.848.3.54 template<typename _Tp , typename _Alloc > void list::unique ( )

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 267 of file list.tcc.

References std::begin(), and std::end().

4.848.3.55 template<typename _Tp , typename _Alloc > template<typename _BinaryPredicate > void list::unique (_BinaryPredicate __*binary_pred*)

Remove consecutive elements satisfying a predicate.

Template Parameters

_*BinaryPredicate*	Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 410 of file list.tcc.

References std::begin(), and std::end().

The documentation for this class was generated from the following files:

- stl_list.h
- list.tcc

4.849 std::locale Class Reference

Classes

- class facet

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

- class id

  Facet ID class.

  The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.

Public Types

- typedef int category

Public Member Functions

- locale () throw ()
- locale (const locale &__other) throw ()
- locale (const char ∗__s)
- locale (const locale &__base, const char ∗__s, category __cat)
- locale (const locale &__base, const locale &__add, category __cat)
- template<typename _Facet >
  locale (const locale &__other, _Facet ∗__f)
- ∼locale () throw ()
- template<typename _Facet >
  locale combine (const locale &__other) const
- string name () const
- bool operator!= (const locale &__other) const throw ()
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool **operator()** (const basic_string< _CharT, _Traits, _Alloc > &__s1, const basic_string< _CharT, _Traits, _Alloc > &__s2) const
- template<typename _Char , typename _Traits , typename _Alloc >
  bool operator() (const basic_string< _Char, _Traits, _Alloc > &__s1, const basic_string< _Char, _Traits, _Alloc > &__s2) const
- const locale & operator= (const locale &__other) throw ()
- bool operator== (const locale &__other) const throw ()

Static Public Member Functions

- static const locale & classic ()
- static locale global (const locale &__loc)

Static Public Attributes

- static const category none
- static const category ctype
- static const category numeric
- static const category collate
- static const category time
- static const category monetary
- static const category messages
- static const category all

Friends

- template<typename _Cache >
  struct **__use_cache**
- class **_Impl**
- class **facet**
- template<typename _Facet >
  bool has_facet (const locale &) throw ()
- template<typename _Facet >
  const _Facet & use_facet (const locale &)

4.849.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file locale_classes.h.

4.849.2 Member Typedef Documentation

4.849.2.1 typedef int **std::locale::category**

Definition of locale::category.

Definition at line 67 of file locale_classes.h.

4.849.3 Constructor & Destructor Documentation

4.849.3.1 std::locale::locale ( ) throw ()

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by combine().

4.849.3.2 std::locale::locale (const **locale** & *__other*) throw ()

Copy constructor.

Constructs a copy of *other*.

Parameters

__other	The locale to copy.

4.849.3.3 std::locale::locale (const char * *__s*) [explicit]

Named locale constructor.

Constructs a copy of the named C library locale.

Parameters

__s	Name of the locale to construct.

Exceptions

std::runtime_error	if __s is null or an undefined locale.

4.849.3.4 std::locale::locale (const locale & *__base*, const char * *__s*, category *__cat*)

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If base is named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__s	Name of the locale to use facets from.
__cat	Set of categories defining the facets to use from __s.

Exceptions

std::runtime_error	if __s is null or an undefined locale.

4.849.3.5 std::locale::locale (const locale & *__base*, const locale & *__add*, category *__cat*)

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__add	The locale to use facets from.
__cat	Set of categories defining the facets to use from add.

4.849.3.6 template<typename _Facet > std::locale::locale (const locale & *__other*, _Facet * *__f*)

Construct locale with another facet.

Constructs a copy of the locale *__other*. The facet *__f* is added to *__other*, replacing an existing facet of type Facet if there is one. If *__f* is null, this locale is a copy of *__other*.

Parameters

__other	The locale to copy.
__f	The facet to add in.

Definition at line 45 of file locale_classes.tcc.

4.849.3.7 std::locale::∼locale ( ) throw ()

Locale destructor.

4.849.4 Member Function Documentation

4.849.4.1 static const locale& std::locale::classic ( ) [static]

Return reference to the C locale.

4.849.4.2 template<typename _Facet > locale std::locale::combine (const locale & __other) const

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type Facet from the locale *other* into the new locale.

Template Parameters

_Facet	The facet type to copy from other

Parameters

__other	The locale to copy from.

Returns

Newly constructed locale.

Exceptions

std::runtime_error	if __other has no facet of type _Facet.

Definition at line 63 of file locale_classes.tcc.

References locale().

4.849.4.3 static locale std::locale::global (const locale & __loc) [static]

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC_ALL, loc.name()).

Parameters

__loc	The new locale to make global.

Returns

Copy of the old global locale.

4.849.4.4 string std::locale::name ( ) const

Return locale name.

Returns

Locale name or "∗" if unnamed.

4.849.4.5 bool std::locale::operator!= (const locale & *__other*) const throw () [inline]

Locale inequality.

Parameters

__other	The locale to compare against.

Returns

! (∗this == __other)

Definition at line 235 of file locale_classes.h.

References operator==().

4.849.4.6 template<typename _Char , typename _Traits , typename _Alloc > bool std::locale::operator() (const basic_string< _Char, _Traits, _Alloc > & *__s1,* const basic_string< _Char, _Traits, _Alloc > & *__s2*) const

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector v of strings could be sorted according to locale loc by doing:

```
std::sort(v.begin(), v.end(), loc);
```

Parameters

__s1	First string to compare.
__s2	Second string to compare.

Returns

True if collate<_Char> facet compares __s1 < __s2, else false.

4.849.4.7 const locale& std::locale::operator= (const locale & *__other*) throw ()

Assignment operator.

Set this locale to be a copy of *other*.

Parameters

__other	The locale to copy.

Returns

A reference to this locale.

4.849.4.8 bool std::locale::operator== (const locale & *__other*) const throw ()

Locale equality.

Parameters

__other	The locale to compare against.

Returns

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

Referenced by operator!=().

4.849.5 Friends And Related Function Documentation

4.849.5.1 template<typename _Facet > bool has_facet (const locale &) throw () [friend]

Test for the presence of a facet.

has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __`loc` contains a facet of type _Facet, else false.

Definition at line 104 of file locale_classes.tcc.

4.849.5.2 template<typename _Facet > const _Facet& use_facet (const locale &) [friend]

Return a facet.

use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

std::bad_cast	if `__loc` doesn't contain a facet of type _Facet.

Definition at line 132 of file locale_classes.tcc.

4.849.6 Member Data Documentation

4.849.6.1 const category std::locale::all `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 105 of file locale_classes.h.

4.849.6.2 const category std::locale::collate `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 101 of file locale_classes.h.

4.849.6.3 const category std::locale::ctype `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 99 of file locale_classes.h.

4.849.6.4 const category std::locale::messages `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 104 of file locale_classes.h.

4.849.6.5 const **category** std::locale::monetary `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 103 of file locale_classes.h.

4.849.6.6 const **category** std::locale::none `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 98 of file locale_classes.h.

4.849.6.7 const **category** std::locale::numeric `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

Definition at line 100 of file locale_classes.h.

4.849.6.8 const **category** std::locale::time `[static]`

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

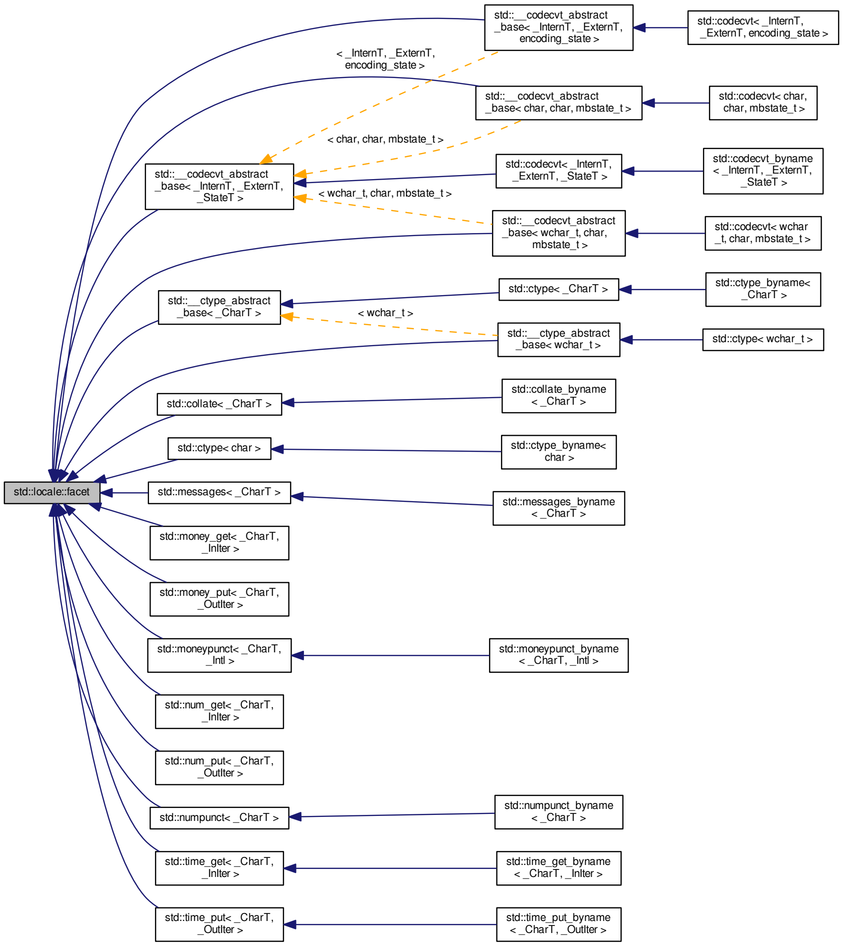
Definition at line 102 of file locale_classes.h.

The documentation for this class was generated from the following files:

- locale_classes.h
- locale_classes.tcc

4.850 std::locale::facet Class Reference

Inheritance diagram for std::locale::facet:



Protected Member Functions

- facet (size_t __refs=0) throw ()
- virtual ∼facet ()

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Friends

- class **locale**
- class **locale::_Impl**

4.850.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

Definition at line 338 of file locale_classes.h.

4.850.2 Constructor & Destructor Documentation

4.850.2.1 std::locale::facet::facet (size_t *__refs* = 0) throw () [inline], [explicit], [protected]

Facet constructor.

This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

Parameters

__refs	The initial value for reference count.

Definition at line 370 of file locale_classes.h.

4.850.2.2 virtual std::locale::facet::∼facet ( ) [protected], [virtual]

Facet destructor.

The documentation for this class was generated from the following file:

- locale_classes.h

4.851 std::locale::id Class Reference

Public Member Functions

- id ()
- size_t **_M_id** () const throw ()

Friends

- template<typename _Facet >
  bool has_facet (const locale &) throw ()
- class **locale**
- class **locale::_Impl**
- template<typename _Facet >
  const _Facet & use_facet (const locale &)

4.851.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.

Definition at line 436 of file locale_classes.h.

4.851.2 Constructor & Destructor Documentation

4.851.2.1 std::locale::id::id ( ) `[inline]`

Constructor.

Definition at line 467 of file locale_classes.h.

4.851.3 Friends And Related Function Documentation

4.851.3.1 template<typename _Facet > bool has_facet (const locale &) throw () `[friend]`

Test for the presence of a facet.

has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if `__loc` contains a facet of type _Facet, else false.

Definition at line 104 of file locale_classes.tcc.

4.851.3.2 template<typename _Facet > const _Facet& use_facet (const locale &) `[friend]`

Return a facet.

use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

std::bad_cast	if __`loc` doesn't contain a facet of type _Facet.

Definition at line 132 of file locale_classes.tcc.

The documentation for this class was generated from the following file:

- locale_classes.h

4.852 std::lock_guard< _Mutex > Class Template Reference

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **lock_guard** (mutex_type &__m)
- **lock_guard** (mutex_type &__m, adopt_lock_t)
- **lock_guard** (const lock_guard &)=delete
- lock_guard & **operator=** (const lock_guard &)=delete

4.852.1 Detailed Description

template<typename _Mutex>class std::lock_guard< _Mutex >

Scoped lock idiom.

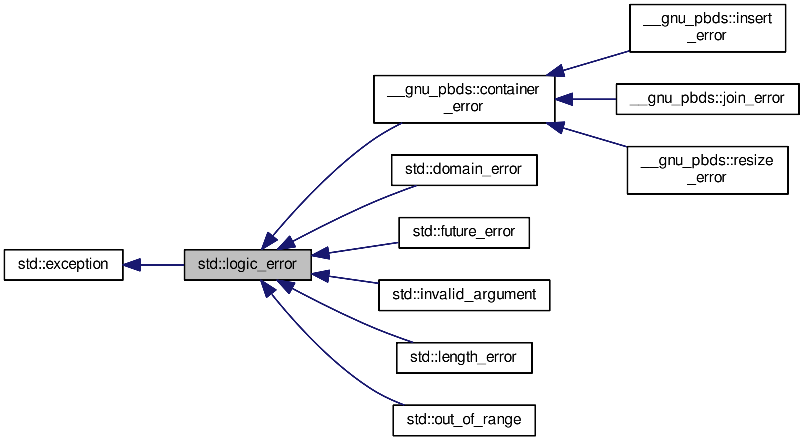
Definition at line 406 of file mutex.

The documentation for this class was generated from the following file:

- mutex

4.853 std::logic_error Class Reference

Inheritance diagram for std::logic_error:



Public Member Functions

- logic_error (const string &__arg)
- virtual const char ∗ what () const noexcept

4.853.1 Detailed Description

One of two subclasses of exception.

Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 55 of file stdexcept.

4.853.2 Constructor & Destructor Documentation

4.853.2.1 std::logic_error::logic_error (const string & *__arg*) `[explicit]`

Takes a character string describing the error.

4.853.3 Member Function Documentation

4.853.3.1 virtual const char∗ std::logic_error::what ( ) const `[virtual]`, `[noexcept]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

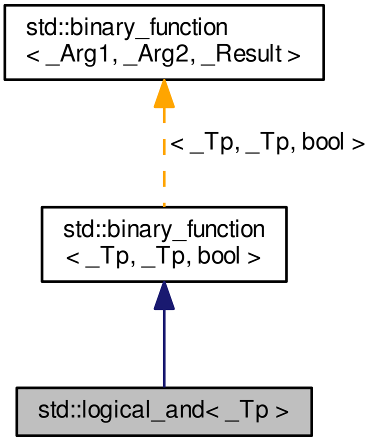
Reimplemented in std::future_error.

The documentation for this class was generated from the following file:

- stdexcept

4.854 std::logical_and< _Tp > Struct Template Reference

Inheritance diagram for std::logical_and< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.854.1 Detailed Description

template<typename _Tp>struct std::logical_and< _Tp >

One of the Boolean operations functors.

Definition at line 268 of file stl_function.h.

4.854.2 Member Typedef Documentation

4.854.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.854.2.2 typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.854.2.3 typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

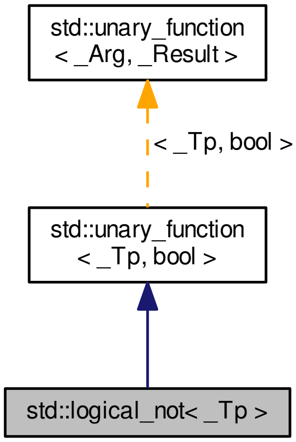
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.855 std::logical_not< _Tp > Struct Template Reference

Inheritance diagram for std::logical_not< _Tp >:



Public Types

- typedef _Tp argument_type
- typedef bool result_type

Public Member Functions

- bool **operator()** (const _Tp &__x) const

4.855.1 Detailed Description

template<typename _Tp>struct std::logical_not< _Tp >

One of the Boolean operations functors.

Definition at line 286 of file stl_function.h.

4.855.2 Member Typedef Documentation

4.855.2.1 typedef _Tp **std::unary_function< _Tp , bool >::argument_type** [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.855.2.2 typedef bool **std::unary_function< _Tp , bool >::result_type** [inherited]

`result_type` is the return type

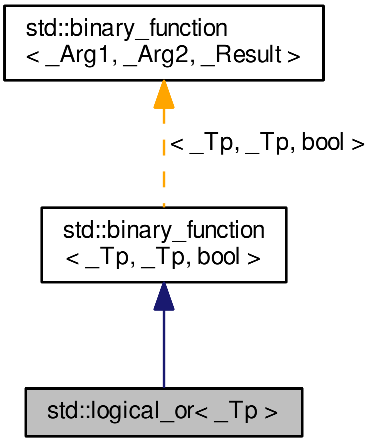
Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.856 std::logical_or< _Tp > Struct Template Reference

Inheritance diagram for std::logical_or< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.856.1 Detailed Description

template<typename _Tp>struct std::logical_or< _Tp >

One of the Boolean operations functors.

Definition at line 277 of file stl_function.h.

4.856.2 Member Typedef Documentation

4.856.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.856.2.2 typedef bool **std::binary_function**< _Tp , _Tp , bool >**::result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.856.2.3 typedef _Tp **std::binary_function**< _Tp , _Tp , bool >**::second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.857 std::lognormal_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **lognormal_distribution** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- **lognormal_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **m** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _RealType **s** () const

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::lognormal_distribution< _RealType1 > &__x)
- bool operator== (const lognormal_distribution &__d1, const lognormal_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::lognormal_distribution< _RealType1 > &__x)

4.857.1 Detailed Description

template<typename _RealType = double>class std::lognormal_distribution< _RealType >

A lognormal_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m,s) = \frac{1}{sx\sqrt{2\pi}} \exp{-\frac{(\ln{x} - m)^2}{2s^2}}$$

Definition at line 2298 of file random.h.

4.857.2 Member Typedef Documentation

4.857.2.1 template<typename _RealType = double> typedef _RealType std::lognormal_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 2301 of file random.h.

4.857.3 Member Function Documentation

4.857.3.1 template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 2389 of file random.h.

References std::numeric_limits< _Tp >::max().

4.857.3.2 template<typename _RealType = double> result_type std::lognormal_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 2382 of file random.h.

4.857.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::lognormal_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 2397 of file random.h.

References std::lognormal_distribution< _RealType >::operator()().

Referenced by std::lognormal_distribution< _RealType >::operator()().

4.857.3.4 template<typename _RealType = double> param_type std::lognormal_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 2367 of file random.h.

4.857.3.5 template<typename _RealType = double> void std::lognormal_distribution< _RealType >::param (const param_type & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 2375 of file random.h.

4.857.3.6 template<typename _RealType = double> void std::lognormal_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 2349 of file random.h.

References std::normal_distribution< _RealType >::reset().

4.857.4 Friends And Related Function Documentation

4.857.4.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & *__os*, const std::lognormal_distribution< _RealType1 > & *__x*) [friend]

Inserts a lognormal_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A lognormal_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.857.4.2 template<typename _RealType = double> bool operator== (const **lognormal_distribution**< _RealType > & *__d1*, const **lognormal_distribution**< _RealType > & *__d2*) `[friend]`

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2434 of file random.h.

4.857.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > **std::basic_istream**<_CharT, _Traits>& operator>> (**std::basic_istream**< _CharT, _Traits > & *__is*, **std::lognormal_distribution**< _RealType1 > & *__x*) `[friend]`

Extracts a lognormal_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A lognormal_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.858 std::lognormal_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef lognormal_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m=_RealType(0), _RealType __s=_RealType(1))
- _RealType **m** () const
- _RealType **s** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.858.1 Detailed Description

template<typename _RealType = double>struct std::lognormal_distribution< _RealType >::param_type

Parameter type.

Definition at line 2307 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.859 std::make_signed< _Tp > Struct Template Reference

Public Types

- typedef __make_signed_selector
  < _Tp >::__type **type**

4.859.1 Detailed Description

template<typename _Tp>struct std::make_signed< _Tp >

make_signed

Definition at line 1612 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.860 std::make_unsigned< _Tp > Struct Template Reference

Public Types

- typedef
  __make_unsigned_selector< _Tp >
  ::__type **type**

4.860.1 Detailed Description

template<typename _Tp>struct std::make_unsigned< _Tp >

make_unsigned

Definition at line 1530 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.861 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Pair_alloc_type::const_pointer **const_pointer**
- typedef _Pair_alloc_type::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Pair_alloc_type::pointer **pointer**
- typedef _Pair_alloc_type::reference **reference**
- typedef _Rep_type::reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- map ()
- map (const _Compare &__comp, const allocator_type &__a=allocator_type())
- map (const map &__x)
- map (map &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- map (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  map (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- mapped_type & at (const key_type &__k)
- const mapped_type & **at** (const key_type &__k) const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- size_type count (const key_type &__x) const
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)

- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > equal_range (const key_type &__x) const
- iterator erase (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const
- allocator_type get_allocator () const noexcept
- std::pair< iterator, bool > insert (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__x)
- void insert (std::initializer_list< value_type > __list)
- iterator insert (const_iterator __position, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- key_compare key_comp () const
- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const
- size_type max_size () const noexcept
- map & operator= (const map &__x)
- map & operator= (map &&__x)
- map & operator= (initializer_list< value_type > __l)
- mapped_type & operator[] (const key_type &__k)
- mapped_type & **operator[]** (key_type &&__k)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (map &__x)
- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const
- value_compare value_comp () const

Friends

- template<typename _K1 , typename _T1 , typename _C1 , typename _A1 >
  bool **operator<** (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)
- template<typename _K1 , typename _T1 , typename _C1 , typename _A1 >
  bool **operator==** (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)

4.861.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>class std::map< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>.

Meets the requirements of a `container`, a `reversible container`, and an `associative container` (using unique keys). For a `map<Key,T>` the key_type is Key, the mapped_type is T, and the value_type is std::pair<const Key,T>.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (∗_unique versus ∗_equal, same as the standard).

Definition at line 96 of file stl_map.h.

4.861.2 Constructor & Destructor Documentation

4.861.2.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map ( ) `[inline]`

Default constructor creates no elements.

Definition at line 160 of file stl_map.h.

4.861.2.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (const _Compare & *__comp,* const allocator_type & *__a* = `allocator_type()`) `[inline]`, `[explicit]`

Creates a map with no elements.

Parameters

__comp	A comparison object.
__a	An allocator object.

Definition at line 169 of file stl_map.h.

4.861.2.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (const map< _Key, _Tp, _Compare, _Alloc > & *__x*) `[inline]`

Map copy constructor.

Parameters

__x	A map of identical element and allocator types.

The newly-created map uses a copy of the allocation object used by *__x*.

Definition at line 180 of file stl_map.h.

4.861.2.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (map< _Key, _Tp, _Compare, _Alloc > && *__x*) [inline], [noexcept]

Map move constructor.

Parameters

__x	A map of identical element and allocator types.

The newly-created map contains the exact contents of *__x*. The contents of *__x* are a valid, but unspecified map.

Definition at line 191 of file stl_map.h.

4.861.2.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::map< _Key, _Tp, _Compare, _Alloc >::map (initializer_list< value_type > *__l*, const _Compare & *__comp* = _Compare(), const allocator_type & *__a* = allocator_type()) [inline]

Builds a map from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison object.
__a	An allocator object.

Create a map consisting of copies of the elements in the initializer_list *__l*. This is linear in N if the range is already sorted, and NlogN otherwise (where N is *__l.size()*).

Definition at line 206 of file stl_map.h.

4.861.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::map< _Key, _Tp, _Compare, _Alloc >::map (_InputIterator *__first*, _InputIterator *__last*) [inline]

Builds a map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a map consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 224 of file stl_map.h.

4.861.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::map< _Key, _Tp, _Compare, _Alloc >::map (_InputIterator *__first*, _InputIterator *__last*, const _Compare & *__comp*, const allocator_type & *__a* = allocator_type()) [inline]

Builds a map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a map consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 241 of file stl_map.h.

4.861.3 Member Function Documentation

4.861.3.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::map< _Key, _Tp, _Compare, _Alloc >::at (const key_type & __k) [inline]

Access to map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equivalent to *__k*, if such a data is present in the map.

Exceptions

std::out_of_range	If no such data is present.

Definition at line 501 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::end(), std::map< _Key, _Tp, _Compare, _Alloc >::key_comp(), and std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound().

4.861.3.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 320 of file stl_map.h.

4.861.3.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 329 of file stl_map.h.

4.861.3.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 393 of file stl_map.h.

4.861.3.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 402 of file stl_map.h.

4.861.3.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 787 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::operator=().

4.861.3.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc >::count (const key_type & __x) const [inline]

Finds the number of elements with given key.

Parameters

__x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 847 of file stl_map.h.

4.861.3.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 411 of file stl_map.h.

4.861.3.9 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 420 of file stl_map.h.

4.861.3.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> std::pair<iterator, bool> std::map< _Key, _Tp, _Compare, _Alloc >::emplace (_Args &&... *__args*) [inline]

Attempts to build and insert a std::pair into the map.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 540 of file stl_map.h.

4.861.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::map< _Key, _Tp, _Compare, _Alloc >::emplace_hint (const_iterator *__pos,* _Args &&... *__args*) [inline]

Attempts to build and insert a std::pair into the map.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args* (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 570 of file stl_map.h.

4.861.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> bool std::map< _Key, _Tp, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the map is empty. (Thus begin() would equal end().)

Definition at line 429 of file stl_map.h.

4.861.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 338 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::at(), and std::map< _Key, _Tp, _Compare, _Alloc >::operator[]().

4.861.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 347 of file stl_map.h.

4.861.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, iterator> std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (const key_type & __x) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 916 of file stl_map.h.

4.861.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator> std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (const key_type & __x) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 935 of file stl_map.h.

4.861.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (const_iterator *__position*) [inline]

Erases an element from a map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 690 of file stl_map.h.

4.861.3.18 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (const key_type & *__x*) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 725 of file stl_map.h.

4.861.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (const_iterator *__first,* const_iterator *__last*) `[inline]`

Erases a [first,last) range of elements from a map.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 745 of file stl_map.h.

4.861.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (const key_type & *__x*) `[inline]`

Tries to locate an element in a map.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 820 of file stl_map.h.

4.861.3.21 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (const key_type & *__x*) const `[inline]`

Tries to locate an element in a map.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 835 of file stl_map.h.

4.861.3.22 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Get a copy of the memory allocation object.

Definition at line 310 of file stl_map.h.

4.861.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, bool> std::map< _Key, _Tp, _Compare, _Alloc >::insert (const value_type & __x) [inline]

Attempts to insert a std::pair into the map.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 594 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::operator=(), and std::map< _Key, _Tp, _Compare, _Alloc >::operator[]().

4.861.3.24 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc >::insert (std::initializer_list< value_type > __list) [inline]

Attempts to insert a list of std::pairs into the map.

Parameters

__list	A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 615 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::insert().

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::insert().

4.861.3.25 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator **std::map**< _Key, _Tp, _Compare, _Alloc >::insert (const_iterator *__position,* const **value_type** & *__x*) [inline]

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 644 of file stl_map.h.

4.861.3.26 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void **std::map**< _Key, _Tp, _Compare, _Alloc >::insert (_InputIterator *__first,* _InputIterator *__last*) [inline]

Template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 670 of file stl_map.h.

4.861.3.27 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> key_compare **std::map**< _Key, _Tp, _Compare, _Alloc >::key_comp ( ) const [inline]

Returns the key comparison object out of which the map was constructed.

Definition at line 796 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::at(), and std::map< _Key, _Tp, _Compare, _Alloc >::operator[]().

4.861.3.28 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator **std::map**< _Key, _Tp, _Compare, _Alloc >::lower_bound (const key_type & *__x*) [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 862 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::at(), and std::map< _Key, _Tp, _Compare, _Alloc >::operator[]().

4.861.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (const key_type & *__x*) const [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 877 of file stl_map.h.

4.861.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the map.

Definition at line 439 of file stl_map.h.

4.861.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (const map< _Key, _Tp, _Compare, _Alloc > & *__x*) [inline]

Map assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

__x	A map of identical element and allocator types.

All the elements of *__x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 264 of file stl_map.h.

4.861.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (map< _Key, _Tp, _Compare, _Alloc > && __x) [inline]

Map move assignment operator.

Parameters

__x	A map of identical element and allocator types.

The contents of *__x* are moved into this map (without copying). *__x* is a valid, but unspecified map.

Definition at line 279 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::clear(), and std::map< _Key, _Tp, _Compare, _Alloc >::swap().

4.861.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> map& std::map< _Key, _Tp, _Compare, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Map list assignment operator.

Parameters

__l	An initializer_list.

This function fills a map with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 300 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::clear(), and std::map< _Key, _Tp, _Compare, _Alloc >::insert().

4.861.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::map< _Key, _Tp, _Compare, _Alloc >::operator[] (const key_type & __k) [inline]

Subscript ([]) access to map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 456 of file stl_map.h.

References std::map< _Key, _Tp, _Compare, _Alloc >::end(), std::map< _Key, _Tp, _Compare, _Alloc >::insert(), std::map< _Key, _Tp, _Compare, _Alloc >::key_comp(), std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound(), and std::piecewise_construct.

4.861.3.35 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 356 of file stl_map.h.

4.861.3.36 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 365 of file stl_map.h.

4.861.3.37 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 374 of file stl_map.h.

4.861.3.38 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 383 of file stl_map.h.

4.861.3.39 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::map< _Key, _Tp, _Compare, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the map.

Definition at line 434 of file stl_map.h.

4.861.3.40 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::map< _Key, _Tp, _Compare, _Alloc >::swap (map< _Key, _Tp, _Compare, _Alloc > & __x) [inline]

Swaps data with another map.

Parameters

__x	A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 777 of file stl_map.h.

Referenced by std::map< _Key, _Tp, _Compare, _Alloc >::operator=(), and std::swap().

4.861.3.41 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (const key_type & __x) [inline]

Finds the end of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 887 of file stl_map.h.

4.861.3.42 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (const key_type & __x) const [inline]

Finds the end of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 897 of file stl_map.h.

4.861.3.43 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> value_compare std::map< _Key, _Tp, _Compare, _Alloc >::value_comp ( ) const [inline]

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 804 of file stl_map.h.

The documentation for this class was generated from the following file:

- stl_map.h

4.862 std::mask_array< _Tp > Class Template Reference

Public Types

- typedef _Tp **value_type**

Public Member Functions

- mask_array (const mask_array &)

- void operator%= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- mask_array & operator= (const mask_array &)
- void **operator=** (const valarray< _Tp > &) const
- void operator= (const _Tp &) const
- template<class _Dom >
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- template<class _Ex >
  void **operator=** (const _Expr< _Ex, _Tp > &__e) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator|=** (const _Expr< _Dom, _Tp > &) const

Friends

- class **valarray< _Tp >**

4.862.1 Detailed Description

template<class _Tp>class std::mask_array< _Tp >

Reference to selected subset of an array.

A mask_array is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a mask_array is to call operator[](valarray<bool>) on a valarray. The returned mask_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if a mask_array is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

Parameters

Tp	Element type.

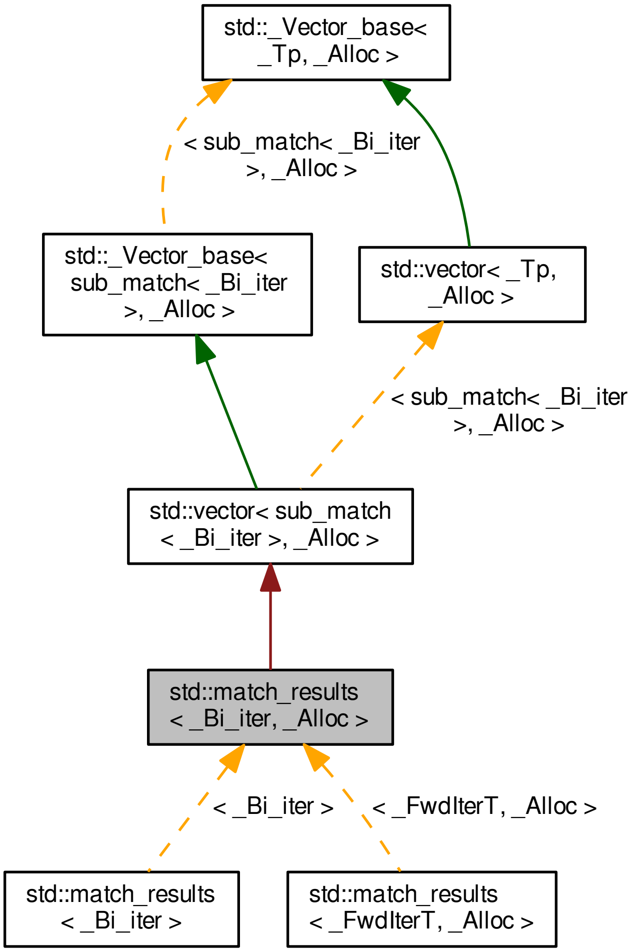
Definition at line 62 of file mask_array.h.

The documentation for this class was generated from the following file:

- mask_array.h

4.863 std::match_results< _Bi_iter, _Alloc > Class Template Reference

Inheritance diagram for std::match_results< _Bi_iter, _Alloc >:



Public Member Functions

- bool ready () const

Private Types

- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef _Base::pointer **pointer**
- typedef std::reverse_iterator < iterator > **reverse_iterator**

Private Member Functions

- pointer **_M_allocate** (size_t __n)
- pointer _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- size_type **_M_check_len** (size_type __n, const char ∗__s) const
- void **_M_deallocate** (pointer __p, size_t __n)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_emplace_back_aux** (_Args &&...__args)
- void **_M_erase_at_end** (pointer __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- void **_M_initialize_dispatch** (_Integer __n, _Integer __value, __true_type)
- void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- void **_M_insert_aux** (iterator __position, _Args &&...__args)
- void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void _M_range_check (size_type __n) const
- void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- bool **_M_shrink_to_fit** ()
- void assign (size_type __n, const value_type &__val)
- void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- reference at (size_type __n)
- const_reference at (size_type __n) const

- reference back ()
- const_reference back () const
- iterator begin () noexcept
- size_type capacity () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- sub_match< _Bi_iter > ∗ data () noexcept
- const sub_match< _Bi_iter > ∗ **data** () const noexcept
- iterator emplace (iterator __position, _Args &&...__args)
- void **emplace_back** (_Args &&...__args)
- iterator end () noexcept
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- reference front ()
- const_reference front () const
- iterator insert (iterator __position, const value_type &__x)
- iterator insert (iterator __position, value_type &&__x)
- void insert (iterator __position, initializer_list< value_type > __l)
- void insert (iterator __position, size_type __n, const value_type &__x)
- void insert (iterator __position, _InputIterator __first, _InputIterator __last)
- reference operator[] (size_type __n)
- const_reference operator[] (size_type __n) const
- void pop_back ()
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- void reserve (size_type __n)
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void shrink_to_fit ()
- void swap (vector &__x) noexcept(_Alloc_traits

Private Attributes

- _Vector_impl **_M_impl**

Friends

- class **__detail::_SpecializedResults< _Bi_iter, _Alloc >**

10.? Public Types

- typedef _Alloc **allocator_type**
- typedef sub_match< _Bi_iter > **value_type**
- typedef const value_type & **const_reference**
- typedef const_reference **reference**
- typedef _Base_type::const_iterator **const_iterator**
- typedef const_iterator **iterator**
- typedef
  __iter_traits::difference_type **difference_type**
- typedef __iter_traits::value_type **char_type**
- typedef allocator_traits
  < _Alloc >::size_type **size_type**
- typedef std::basic_string
  < char_type > **string_type**

28.10.1 Construction, Copying, and Destruction

- match_results (const _Alloc &__a=_Alloc())
- match_results (const match_results &__rhs)
- match_results (match_results &&__rhs) noexcept
- match_results & operator= (const match_results &__rhs)
- match_results & operator= (match_results &&__rhs)
- ∼match_results ()

28.10.2 Size

- size_type size () const
- size_type max_size () const
- bool empty () const

10.3 Element Access

- difference_type length (size_type __sub=0) const
- difference_type position (size_type __sub=0) const
- string_type str (size_type __sub=0) const
- const_reference operator[] (size_type __sub) const
- const_reference prefix () const
- const_reference suffix () const
- const_iterator begin () const
- const_iterator cbegin () const
- const_iterator end () const
- const_iterator cend () const

10.4 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- template<typename _Out_iter >
  _Out_iter format (_Out_iter __out, const char_type ∗__fmt_first, const char_type ∗__fmt_last, match_flag_type __flags=regex_constants::format_default) const
- template<typename _Out_iter , typename _St , typename _Sa >
  _Out_iter format (_Out_iter __out, const basic_string< char_type, _St, _Sa > &__fmt, match_flag_type __flags=regex_constants::format_default) const
- template<typename _Out_iter , typename _St , typename _Sa >
  basic_string< char_type, _St, _Sa > format (const basic_string< char_type, _St, _Sa > &__fmt, match_flag_type __flags=regex_constants::format_default) const
- string_type format (const char_type ∗__fmt, match_flag_type __flags=regex_constants::format_default) const

10.5 Allocator

- allocator_type get_allocator () const

10.6 Swap

- void swap (match_results &__that)

4.863.1 Detailed Description

template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>class std::match_results< _Bi_iter, _Alloc >

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template match_results.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The sub_match object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the sub_match member matched is always true. The sub_match object stored at index n denotes what matched the marked sub-expression n within the matched expression. If the sub-expression n participated in a regular expression match then the sub_match member matched evaluates to true, and members first and second denote the range of characters [first, second) which formed that match. Otherwise matched is false, and members first and second point to the end of the sequence that was searched.

Definition at line 1429 of file regex.h.

4.863.2 Constructor & Destructor Documentation

4.863.2.1 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter, _Alloc >::match_results (const _Alloc & *__a* = `_Alloc()`) `[inline]`, `[explicit]`

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

Definition at line 1478 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::operator=().

4.863.2.2 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter, _Alloc >::match_results (const match_results< _Bi_iter, _Alloc > & __rhs) [inline]

Copy constructs a match_results.

Definition at line 1485 of file regex.h.

4.863.2.3 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter, _Alloc >::match_results (match_results< _Bi_iter, _Alloc > && __rhs) [inline], [noexcept]

Move constructs a match_results.

Definition at line 1492 of file regex.h.

4.863.2.4 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> std::match_results< _Bi_iter, _Alloc >::~match_results ( ) [inline]

Destroys a match_results object.

Definition at line 1519 of file regex.h.

4.863.3 Member Function Documentation

4.863.3.1 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator std::match_results< _Bi_iter, _Alloc >::begin ( ) const [inline]

Gets an iterator to the start of the sub_match collection.

Definition at line 1676 of file regex.h.

Referenced by std::operator==().

4.863.3.2 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator std::match_results< _Bi_iter, _Alloc >::cbegin ( ) const [inline]

Gets an iterator to the start of the sub_match collection.

Definition at line 1683 of file regex.h.

4.863.3.3 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator std::match_results< _Bi_iter, _Alloc >::cend ( ) const [inline]

Gets an iterator to one-past-the-end of the collection.

Definition at line 1697 of file regex.h.

4.863.3.4 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> bool std::match_results< _Bi_iter, _Alloc >::empty ( ) const [inline]

Indicates if the match_results contains no results.

Return values

true	The match_results object is empty.
false	The match_results object is not empty.

Definition at line 1563 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::end(), std::operator==(), std::match_results< _FwdIterT, _Alloc >::prefix(), and std::match_results< _FwdIterT, _Alloc >::suffix().

4.863.3.5 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_iterator **std::match_results**< _Bi_iter, _Alloc >::end ( ) const `[inline]`

Gets an iterator to one-past-the-end of the collection.

Definition at line 1690 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::cend(), and std::operator==().

4.863.3.6 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter > _Out_iter **std::match_results**< _Bi_iter, _Alloc >::format (_Out_iter *__out,* const char_type ∗ *__fmt_first,* const char_type ∗ *__fmt_last,* match_flag_type *__flags* = **regex_constants::format_default**) const `[inline]`

Precondition

ready() == true

Todo Implement this function.

Definition at line 1718 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::format().

4.863.3.7 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter , typename _St , typename _Sa > _Out_iter **std::match_results**< _Bi_iter, _Alloc >::format (_Out_iter *__out,* const **basic_string**< char_type, _St, _Sa > & *__fmt,* match_flag_type *__flags* = **regex_constants::format_default**) const `[inline]`

Precondition

ready() == true

Definition at line 1728 of file regex.h.

4.863.3.8 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> template<typename _Out_iter , typename _St , typename _Sa > **basic_string**<char_type, _St, _Sa> **std::match_results**< _Bi_iter, _Alloc >::format (const **basic_string**< char_type, _St, _Sa > & *__fmt,* match_flag_type *__flags* = **regex_constants::format_default**) const `[inline]`

Precondition

ready() == true

Definition at line 1740 of file regex.h.

4.863.3.9 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> **string_type** **std::match_results**< _Bi_iter, _Alloc >::format (const char_type ∗ *__fmt,* match_flag_type *__flags* = **regex_constants::format_default**) const `[inline]`

Precondition

ready() == true

Definition at line 1752 of file regex.h.

4.863.3.10 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> allocator_type std::match_results< _Bi_iter, _Alloc >::get_allocator ( ) const [inline]

Gets a copy of the allocator.

Definition at line 1773 of file regex.h.

4.863.3.11 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type std::match_results< _Bi_iter, _Alloc >::length (size_type *__sub* = 0) const [inline]

Gets the length of the indicated submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

Definition at line 1582 of file regex.h.

4.863.3.12 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> size_type std::match_results< _Bi_iter, _Alloc >::max_size ( ) const [inline]

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Definition at line 1554 of file regex.h.

4.863.3.13 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results& std::match_results< _Bi_iter, _Alloc >::operator= (const match_results< _Bi_iter, _Alloc > & *__rhs*) [inline]

Assigns rhs to ∗this.

Definition at line 1500 of file regex.h.

4.863.3.14 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> match_results& std::match_results< _Bi_iter, _Alloc >::operator= (match_results< _Bi_iter, _Alloc > && *__rhs*) [inline]

Move-assigns rhs to ∗this.

Definition at line 1510 of file regex.h.

4.863.3.15 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference std::match_results< _Bi_iter, _Alloc >::operator[] (size_type *__sub*) const [inline]

Gets a sub_match reference for the match or submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function gets a reference to the indicated submatch, or the entire match if ___`sub` is zero.

If ___`sub` >= size() then this function returns a sub_match with a special value indicating no submatch.

Definition at line 1630 of file regex.h.

4.863.3.16 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> difference_type std::match_results< _Bi_iter, _Alloc >::position (size_type *__sub* = 0) const [inline]

Gets the offset of the beginning of the indicated submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of ___`sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Returns -1 if ___`sub` is out of range.

Definition at line 1599 of file regex.h.

4.863.3.17 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference std::match_results< _Bi_iter, _Alloc >::prefix ( ) const [inline]

Gets a sub_match representing the match prefix.

Precondition

ready() == true

This function gets a reference to a sub_match object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1647 of file regex.h.

Referenced by std::operator==(), and std::match_results< _FwdIterT, _Alloc >::position().

4.863.3.18 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> bool std::match_results< _Bi_iter, _Alloc >::ready ( ) const [inline]

Indicates if the match_results is ready.

Return values

true	The object has a fully-established result state.
false	The object is not ready.

Definition at line 1530 of file regex.h.

Referenced by std::operator==(), std::match_results< _FwdIterT, _Alloc >::operator[ ](), std::match_results< _FwdIterT, _Alloc >::prefix(), and std::match_results< _FwdIterT, _Alloc >::suffix().

4.863.3.19 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> size_type std::match_results< _Bi_iter, _Alloc >::size ( ) const [inline]

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Definition at line 1547 of file regex.h.

Referenced by std::match_results< _FwdIterT, _Alloc >::empty(), std::operator==(), std::match_results< _FwdIterT, _Alloc >::operator[ ](), and std::match_results< _FwdIterT, _Alloc >::position().

4.863.3.20 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> string_type std::match_results< _Bi_iter, _Alloc >::str (size_type *__sub* = 0) const [inline]

Gets the match or submatch converted to a string type.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

Definition at line 1615 of file regex.h.

4.863.3.21 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> const_reference std::match_results< _Bi_iter, _Alloc >::suffix ( ) const [inline]

Gets a sub_match representing the match suffix.

Precondition

ready() == true

This function gets a reference to a sub_match object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1664 of file regex.h.

Referenced by std::operator==().

4.863.3.22 template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >> void **std::match_results**< _Bi_iter, _Alloc >::swap (**match_results**< _Bi_iter, _Alloc > & *__that*) [inline]

Swaps the contents of two match_results.

Definition at line 1787 of file regex.h.

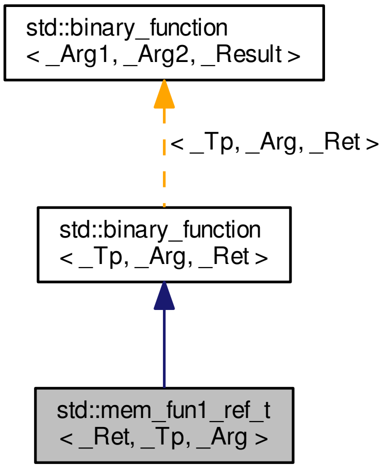
Referenced by std::swap().

The documentation for this class was generated from the following file:

- regex.h

4.864 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

Inheritance diagram for std::mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **mem_fun1_ref_t** (_Ret(_Tp::∗__pf)(_Arg))
- _Ret **operator()** (_Tp &__r, _Arg __x) const

4.864.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

Definition at line 650 of file stl_function.h.

4.864.2 Member Typedef Documentation

4.864.2.1 typedef _Tp std::binary_function< _Tp , _Arg , _Ret >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.864.2.2 typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.864.2.3 typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

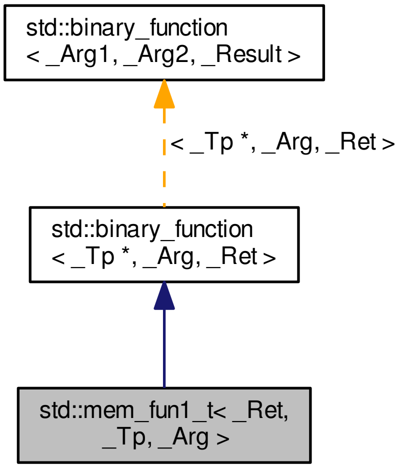
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.865 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

Inheritance diagram for std::mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp ∗ first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **mem_fun1_t** (_Ret(_Tp::∗__pf)(_Arg))
- _Ret **operator()** (_Tp ∗__p, _Arg __x) const

4.865.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>class std::mem_fun1_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

Definition at line 614 of file stl_function.h.

4.865.2 Member Typedef Documentation

4.865.2.1 typedef _Tp ∗ std::binary_function< _Tp ∗ , _Arg , _Ret >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.865.2.2 typedef _Ret **std::binary_function**< _Tp ∗ , _Arg , _Ret >::**result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.865.2.3 typedef _Arg **std::binary_function**< _Tp ∗ , _Arg , _Ret >::**second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

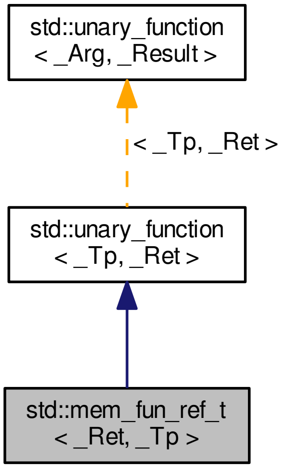
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.866 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference

Inheritance diagram for std::mem_fun_ref_t< _Ret, _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Ret result_type

Public Member Functions

- **mem_fun_ref_t** (_Ret(_Tp::∗__pf)())
- _Ret **operator()** (_Tp &__r) const

4.866.1 Detailed Description

template<typename _Ret, typename _Tp>class std::mem_fun_ref_t< _Ret, _Tp >

One of the adaptors for member pointers.

Definition at line 578 of file stl_function.h.

4.866.2 Member Typedef Documentation

4.866.2.1 typedef _Tp std::unary_function< _Tp , _Ret >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.866.2.2 typedef _Ret std::unary_function< _Tp , _Ret >::result_type [inherited]

`result_type` is the return type

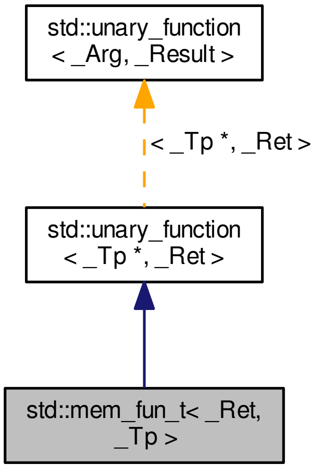
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.867 std::mem_fun_t< _Ret, _Tp > Class Template Reference

Inheritance diagram for std::mem_fun_t< _Ret, _Tp >:



Public Types

- typedef _Tp ∗ argument_type
- typedef _Ret result_type

Public Member Functions

- **mem_fun_t** (_Ret(_Tp::∗__pf)())
- _Ret **operator()** (_Tp ∗__p) const

4.867.1 Detailed Description

template<typename _Ret, typename _Tp>class std::mem_fun_t< _Ret, _Tp >

One of the adaptors for member pointers.

Definition at line 542 of file stl_function.h.

4.867.2 Member Typedef Documentation

4.867.2.1 typedef _Tp ∗ std::unary_function< _Tp ∗ , _Ret >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.867.2.2 typedef _Ret std::unary_function< _Tp ∗ , _Ret >::result_type [inherited]

`result_type` is the return type

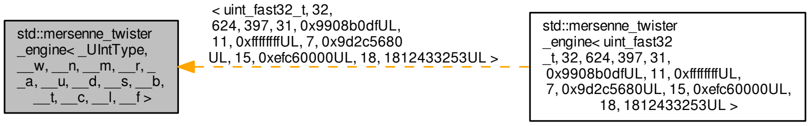
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.868 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference

Inheritance diagram for std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >:



Public Types

- typedef _UIntType result_type

Public Member Functions

- **mersenne_twister_engine** (result_type __sd=default_seed)
- template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value> ::type>
  mersenne_twister_engine (_Sseq &__q)
- void discard (unsigned long long __z)
- result_type **operator()** ()
- void **seed** (result_type __sd=default_seed)
- template<typename _Sseq >
  std::enable_if< std::is_class
  < _Sseq >::value >::type **seed** (_Sseq &__q)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr result_type **initialization_multiplier**
- static constexpr size_t **mask_bits**
- static constexpr size_t **shift_size**
- static constexpr size_t **state_size**
- static constexpr result_type **tempering_b**
- static constexpr result_type **tempering_c**
- static constexpr result_type **tempering_d**
- static constexpr size_t **tempering_l**
- static constexpr size_t **tempering_s**
- static constexpr size_t **tempering_t**
- static constexpr size_t **tempering_u**
- static constexpr size_t **word_size**
- static constexpr result_type **xor_mask**

Friends

- template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)
- bool operator== (const mersenne_twister_engine &__lhs, const mersenne_twister_engine &__rhs)
- template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)

4.868.1 Detailed Description

template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined mt19937 class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

Template Parameters

__w	Word size, the number of bits in each element of the state vector.
__n	The degree of recursion.
__m	The period parameter.
__r	The separation point bit index.
__a	The last row of the twist matrix.
__u	The first right-shift tempering matrix parameter.
__d	The first right-shift tempering matrix mask.
__s	The first left-shift tempering matrix parameter.
__b	The first left-shift tempering matrix mask.
__t	The second left-shift tempering matrix parameter.
__c	The second left-shift tempering matrix mask.
__l	The second right-shift tempering matrix parameter.
__f	Initialization multiplier.

Definition at line 449 of file random.h.

4.868.2 Member Typedef Documentation

4.868.2.1 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type

The type of the generated random value.

Definition at line 452 of file random.h.

4.868.3 Constructor & Destructor Documentation

4.868.3.1 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, mersenne_twister_engine>::value> ::type> std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine (_Sseq & *__q*) `[inline]`, `[explicit]`

Constructs a mersenne_twister_engine random number generator engine seeded from the seed sequence `__q`.

Parameters

__q	the seed sequence.

Definition at line 513 of file random.h.

4.868.4 Member Function Documentation

4.868.4.1 template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::discard (unsigned long long *__z*)

Discard a sequence of random numbers.

Definition at line 435 of file bits/random.tcc.

4.868.4.2 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::max ( ) `[inline]`, `[static]`

Gets the largest possible value in the output range.

Definition at line 534 of file random.h.

4.868.4.3 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::min ( ) `[inline]`, `[static]`

Gets the smallest possible value in the output range.

Definition at line 527 of file random.h.

4.868.5 Friends And Related Function Documentation

4.868.5.1 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & *__os*, const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & *__x*) `[friend]`

Inserts the current state of a % mersenne_twister_engine random number generator engine `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A % mersenne_twister_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.868.5.2 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> bool operator== (const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & *__lhs*, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & *__rhs*) [friend]

Compares two % mersenne_twister_engine random number generator objects of the same type for equality.

Parameters

__lhs	A % mersenne_twister_engine random number generator object.
__rhs	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 559 of file random.h.

4.868.5.3 template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f> template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & *__is*, std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & *__x*) [friend]

Extracts the current state of a % mersenne_twister_engine random number generator engine ___x from the input stream ___is.

Parameters

__is	An input stream.
__x	A % mersenne_twister_engine random number generator engine.

Returns

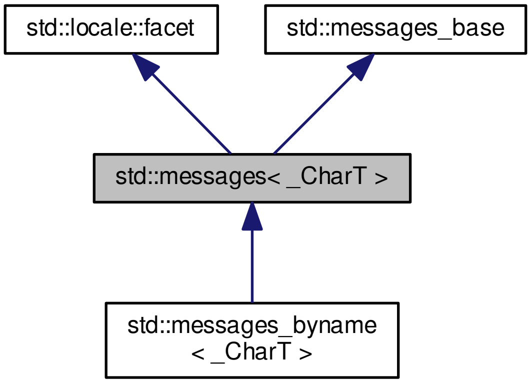
The input stream with the state of ___x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.869 std::messages< _CharT > Class Template Reference

Inheritance diagram for std::messages< _CharT >:



Public Types

- typedef int **catalog**

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- messages (size_t __refs=0)
- messages (__c_locale __cloc, const char ∗__s, size_t __refs=0)
- void **close** (catalog __c) const
- string_type **get** (catalog __c, int __set, int __msgid, const string_type &__s) const
- catalog **open** (const basic_string< char > &__s, const locale &__loc) const
- catalog **open** (const basic_string< char > &, const locale &, const char ∗) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼messages ()
- string_type **_M_convert_from_char** (char ∗) const
- char ∗ **_M_convert_to_char** (const string_type &__msg) const
- virtual void **do_close** (catalog) const

- virtual string_type **do_get** (catalog, int, int, const string_type &__dfault) const
- template<>
  string **do_get** (catalog, int, int, const string &) const
- template<>
  wstring **do_get** (catalog, int, int, const wstring &) const
- virtual catalog **do_open** (const basic_string< char > &, const locale &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char ∗ **_M_name_messages**

4.869.1 Detailed Description

template<typename _CharT>class std::messages< _CharT >

Primary class template messages.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar_t instantiations.

The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

Definition at line 1695 of file locale_facets_nonio.h.

4.869.2 Member Typedef Documentation

4.869.2.1 template<typename _CharT > typedef _CharT std::messages< _CharT >::char_type

Public typedefs.

Definition at line 1701 of file locale_facets_nonio.h.

4.869.2.2 template<typename _CharT > typedef basic_string<_CharT> std::messages< _CharT >::string_type

Public typedefs.

Definition at line 1702 of file locale_facets_nonio.h.

4.869.3 Constructor & Destructor Documentation

4.869.3.1 template<typename _CharT > std::messages< _CharT >::messages (size_t __refs = 0) [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 44 of file messages_members.h.

4.869.3.2 template<typename _CharT > std::messages< _CharT >::messages (__c_locale __cloc, const char * __s, size_t __refs = 0) [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__s	The name of a locale.
__refs	Refcount to pass to the base class.

Definition at line 50 of file messages_members.h.

4.869.3.3 template<typename _CharT > std::messages< _CharT >::∼messages ( ) [protected], [virtual]

Destructor.

Definition at line 79 of file messages_members.h.

4.869.4 Member Data Documentation

4.869.4.1 template<typename _CharT > locale::id std::messages< _CharT >::id [static]

Numpunct facet id.

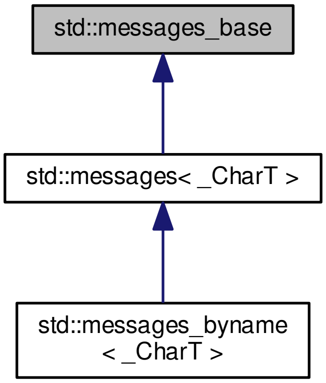
Definition at line 1713 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- messages_members.h

4.870 std::messages_base Struct Reference

Inheritance diagram for std::messages_base:



Public Types

- typedef int **catalog**

4.870.1 Detailed Description

Messages facet base class providing catalog typedef.

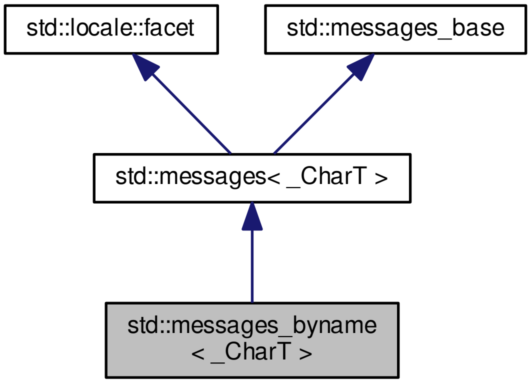
Definition at line 1668 of file locale_facets_nonio.h.

The documentation for this struct was generated from the following file:

- locale_facets_nonio.h

4.871 std::messages_byname< _CharT > Class Template Reference

Inheritance diagram for std::messages_byname< _CharT >:



Public Types

- typedef int **catalog**
- typedef _CharT **char_type**
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **messages_byname** (const char ∗__s, size_t __refs=0)
- void **close** (catalog __c) const
- string_type **get** (catalog __c, int __set, int __msgid, const string_type &__s) const
- catalog **open** (const basic_string< char > &__s, const locale &__loc) const
- catalog **open** (const basic_string< char > &, const locale &, const char ∗) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- string_type **_M_convert_from_char** (char ∗) const
- char ∗ **_M_convert_to_char** (const string_type &__msg) const
- virtual void **do_close** (catalog) const
- virtual string_type **do_get** (catalog, int, int, const string_type &__dfault) const
- template<>
  string **do_get** (catalog, int, int, const string &) const

- template<>
  wstring **do_get** (catalog, int, int, const wstring &) const
- virtual catalog **do_open** (const basic_string< char > &, const locale &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char ∗ **_M_name_messages**

4.871.1 Detailed Description

template<typename _CharT>class std::messages_byname< _CharT >

class messages_byname [22.2.7.2].

Definition at line 1879 of file locale_facets_nonio.h.

4.871.2 Member Data Documentation

4.871.2.1 template<typename _CharT > locale::id std::messages< _CharT >::id `[static]`, `[inherited]`

Numpunct facet id.

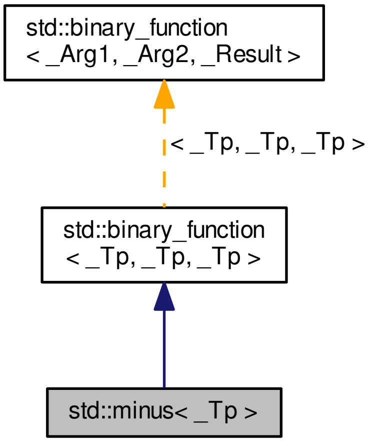
Definition at line 1713 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- messages_members.h

4.872 std::minus< _Tp > Struct Template Reference

Inheritance diagram for std::minus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

4.872.1 Detailed Description

template<typename _Tp>struct std::minus< _Tp >

One of the math functors.

Definition at line 149 of file stl_function.h.

4.872.2 Member Typedef Documentation

4.872.2.1 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.872.2.2 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.872.2.3 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type `[inherited]`

`second_argument_type` is the type of the second argument

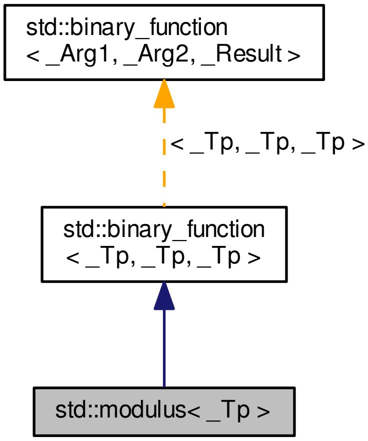
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.873 std::modulus< _Tp > Struct Template Reference

Inheritance diagram for std::modulus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

4.873.1 Detailed Description

template<typename _Tp>struct std::modulus< _Tp >

One of the math functors.

Definition at line 176 of file stl_function.h.

4.873.2 Member Typedef Documentation

4.873.2.1 typedef _Tp **std::binary_function< _Tp , _Tp , _Tp >::first_argument_type** `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.873.2.2 typedef _Tp **std::binary_function< _Tp , _Tp , _Tp >::result_type** `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.873.2.3 typedef _Tp **std::binary_function< _Tp , _Tp , _Tp >::second_argument_type** `[inherited]`

`second_argument_type` is the type of the second argument

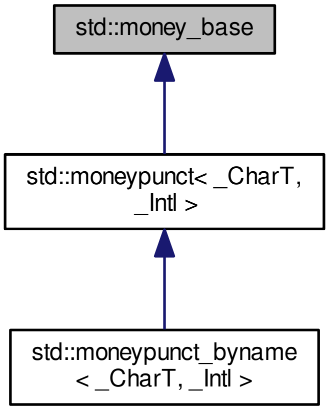
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.874 std::money_base Class Reference

Inheritance diagram for std::money_base:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- enum **part** {
  none, **space**, **symbol**, **sign**,
  value }

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char ∗ **_S_atoms**
- static const pattern **_S_default_pattern**

4.874.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See Also

moneypunct::pos_format() and moneypunct::neg_format() for details of how these fields are interpreted.

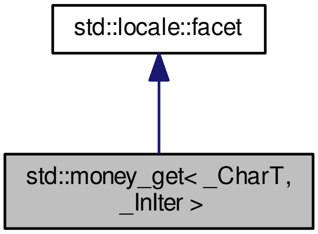
Definition at line 840 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

4.875 std::money_get< _CharT, _InIter > Class Template Reference

Inheritance diagram for std::money_get< _CharT, _InIter >:



Public Types

- typedef _CharT char_type
- typedef _InIter iter_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- money_get (size_t __refs=0)
- iter_type get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const
- iter_type get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼money_get ()
- template<bool _Intl>
  iter_type **_M_extract** (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits) const
- virtual iter_type do_get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const
- virtual iter_type do_get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.875.1 Detailed Description

template<typename _CharT, typename _InIter>class std::money_get< _CharT, _InIter >

Primary class template money_get.

This facet encapsulates the code to parse and return a monetary amount from a string.

The money_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_get facet.

Definition at line 1370 of file locale_facets_nonio.h.

4.875.2 Member Typedef Documentation

4.875.2.1 template<typename _CharT , typename _InIter > typedef _CharT **std::money_get**< _CharT, _InIter >::**char_type**

Public typedefs.

Definition at line 1376 of file locale_facets_nonio.h.

4.875.2.2 template<typename _CharT , typename _InIter > typedef _InIter **std::money_get**< _CharT, _InIter >::**iter_type**

Public typedefs.

Definition at line 1377 of file locale_facets_nonio.h.

4.875.2.3 template<typename _CharT , typename _InIter > typedef **basic_string**<_CharT> **std::money_get**< _CharT, _InIter >::**string_type**

Public typedefs.

Definition at line 1378 of file locale_facets_nonio.h.

4.875.3 Constructor & Destructor Documentation

4.875.3.1 template<typename _CharT , typename _InIter > **std::money_get**< _CharT, _InIter >::**money_get** (size_t *__refs* = 0) `[inline]`, `[explicit]`

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 1392 of file locale_facets_nonio.h.

4.875.3.2 template<typename _CharT , typename _InIter > virtual **std::money_get**< _CharT, _InIter >::~**money_get** ( ) `[inline]`, `[protected]`, `[virtual]`

Destructor.

Definition at line 1460 of file locale_facets_nonio.h.

4.875.4 Member Function Documentation

4.875.4.1 template<typename _CharT , typename _InIter > _InIter **std::money_get**< _CharT, _InIter >::do_get (**iter_type** *__s*, **iter_type** *__end*, bool *__intl*, **ios_base** & *__io*, **ios_base::iostate** & *__err*, long double & *__units*) const `[protected]`, `[virtual]`

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See Also

get() for details.

Definition at line 365 of file locale_facets_nonio.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str().

Referenced by std::money_get< _CharT, _InIter >::get().

4.875.4.2 template<typename _CharT , typename _InIter > _InIter std::money_get< _CharT, _InIter >::do_get (iter_type *__s,* iter_type *__end,* bool *__intl,* ios_base & *__io,* ios_base::iostate & *__err,* string_type & *__digits*) const [protected], [virtual]

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See Also

get() for details.

Definition at line 378 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::basic_string< _CharT, _Traits, _Alloc >::resize(), and std::__ctype_abstract_base< _CharT >::widen().

4.875.4.3 template<typename _CharT , typename _InIter > iter_type std::money_get< _CharT, _InIter >::get (iter_type *__s,* iter_type *__end,* bool *__intl,* ios_base & *__io,* ios_base::iostate & *__err,* long double & *__units*) const [inline]

Read and parse a monetary value.

This function reads characters from *__s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *units* as an integral value moneypunct::frac_digits() ∗ the actual amount. For example, the string $10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit). *units* is unchanged if parsing fails.

This function works by returning the result of do_get().

Parameters

__s	Start of characters to parse.
__end	End of characters to parse.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__err	Error field to set if parsing fails.
__units	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1422 of file locale_facets_nonio.h.

References std::money_get< _CharT, _InIter >::do_get().

4.875.4.4 template<typename _CharT , typename _InIter > **iter_type std::money_get< _CharT, _InIter >::get** (**iter_type** *__s,* **iter_type** *__end,* bool *__intl,* **ios_base &** *__io,* **ios_base::iostate &** *__err,* **string_type &** *__digits*) const `[inline]`

Read and parse a monetary value.

This function reads characters from *__s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *digits*. For example, the string $10.01 in a US locale would store `1001` in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit).

This function works by returning the result of do_get().

Parameters

__s	Start of characters to parse.
__end	End of characters to parse.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__err	Error field to set if parsing fails.
__digits	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1453 of file locale_facets_nonio.h.

References std::money_get< _CharT, _InIter >::do_get().

4.875.5 Member Data Documentation

4.875.5.1 template<typename _CharT , typename _InIter > **locale::id std::money_get< _CharT, _InIter >::id** `[static]`

Numpunct facet id.

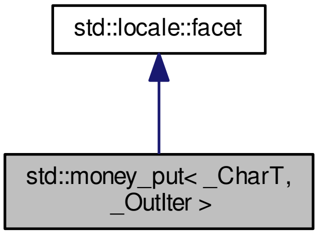
Definition at line 1382 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

4.876 std::money_put< _CharT, _OutIter > Class Template Reference

Inheritance diagram for std::money_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- money_put (size_t __refs=0)
- iter_type put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const
- iter_type put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼money_put ()
- template<bool _Intl>
  iter_type **_M_insert** (iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const
- virtual iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const
- virtual iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)

- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.876.1 Detailed Description

template<typename _CharT, typename _OutIter>class std::money_put< _CharT, _OutIter >

Primary class template money_put.

This facet encapsulates the code to format and output a monetary amount.

The money_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_put facet.

Definition at line 1521 of file locale_facets_nonio.h.

4.876.2 Member Typedef Documentation

4.876.2.1 template<typename _CharT , typename _OutIter > typedef _CharT std::money_put< _CharT, _OutIter >::char_type

Public typedefs.

Definition at line 1526 of file locale_facets_nonio.h.

4.876.2.2 template<typename _CharT , typename _OutIter > typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type

Public typedefs.

Definition at line 1527 of file locale_facets_nonio.h.

4.876.2.3 template<typename _CharT , typename _OutIter > typedef basic_string<_CharT> std::money_put< _CharT, _OutIter >::string_type

Public typedefs.

Definition at line 1528 of file locale_facets_nonio.h.

4.876.3 Constructor & Destructor Documentation

4.876.3.1 template<typename _CharT , typename _OutIter > std::money_put< _CharT, _OutIter >::money_put (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 1542 of file locale_facets_nonio.h.

4.876.3.2 template<typename _CharT , typename _OutIter > virtual **std::money_put**< _CharT, _OutIter >::∼**money_put** ( ) `[inline]`, `[protected]`, `[virtual]`

Destructor.

Definition at line 1592 of file locale_facets_nonio.h.

4.876.4 Member Function Documentation

4.876.4.1 template<typename _CharT , typename _OutIter > _OutIter **std::money_put**< _CharT, _OutIter >::do_put (**iter_type** *__s*, bool *__intl*, **ios_base** & *__io*, **char_type** *__fill*, long double *__units*) const `[protected]`, `[virtual]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the value 1001 in a US locale would write `$10.01` to *__s*.

This function is a hook for derived classes to change the value returned.

See Also

put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__units	Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 570 of file locale_facets_nonio.tcc.

References std::ios_base::getloc(), and std::__ctype_abstract_base< _CharT >::widen().

Referenced by std::money_put< _CharT, _OutIter >::put().

4.876.4.2 template<typename _CharT , typename _OutIter > _OutIter **std::money_put**< _CharT, _OutIter >::do_put (**iter_type** *__s*, bool *__intl*, **ios_base** & *__io*, **char_type** *__fill*, const **string_type** & *__digits*) const `[protected]`, `[virtual]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the string `1001` in a US locale would write `$10.01` to *__s*.

This function is a hook for derived classes to change the value returned.

See Also

put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__digits	Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 608 of file locale_facets_nonio.tcc.

4.876.4.3 template<typename _CharT , typename _OutIter > iter_type std::money_put< _CharT, _OutIter >::put (iter_type *__s,* bool *__intl,* ios_base & *__io,* char_type *__fill,* long double *__units*) const `[inline]`

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to __*s*. For example, the value 1001 in a US locale would write `$10.01` to __*s*.

This function works by returning the result of do_put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__units	Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1562 of file locale_facets_nonio.h.

References std::money_put< _CharT, _OutIter >::do_put().

4.876.4.4 template<typename _CharT , typename _OutIter > iter_type std::money_put< _CharT, _OutIter >::put (iter_type *__s,* bool *__intl,* ios_base & *__io,* char_type *__fill,* const string_type & *__digits*) const `[inline]`

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to __*s*. For example, the string `1001` in a US locale would write `$10.01` to __*s*.

This function works by returning the result of do_put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__digits	Place to store result of parsing.

Returns

Iterator after writing.

Definition at line 1585 of file locale_facets_nonio.h.

References std::money_put< _CharT, _OutIter >::do_put().

4.876.5 Member Data Documentation

4.876.5.1 template<typename _CharT , typename _OutIter > locale::id std::money_put< _CharT, _OutIter >::id `[static]`

Numpunct facet id.

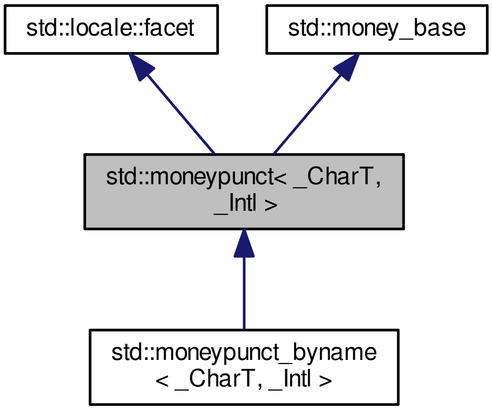
Definition at line 1532 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

4.877 std::moneypunct< _CharT, _Intl > Class Template Reference

Inheritance diagram for std::moneypunct< _CharT, _Intl >:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- typedef __moneypunct_cache < _CharT, _Intl > **__cache_type**
- enum **part** { **none**, **space**, **symbol**, **sign**, **value** }

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- moneypunct (size_t __refs=0)
- moneypunct (__cache_type ∗__cache, size_t __refs=0)
- moneypunct (__c_locale __cloc, const char ∗__s, size_t __refs=0)
- string_type curr_symbol () const
- char_type decimal_point () const
- int frac_digits () const
- string grouping () const
- string_type negative_sign () const
- string_type positive_sign () const
- char_type thousands_sep () const

- pattern pos_format () const
- pattern neg_format () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char ∗ **_S_atoms**
- static const pattern **_S_default_pattern**
- static locale::id id
- static const bool intl

Protected Member Functions

- virtual ∼moneypunct ()
- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char ∗__name=0)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- virtual string_type do_curr_symbol () const
- virtual char_type do_decimal_point () const
- virtual int do_frac_digits () const
- virtual string do_grouping () const
- virtual pattern do_neg_format () const
- virtual string_type do_negative_sign () const
- virtual pattern do_pos_format () const
- virtual string_type do_positive_sign () const
- virtual char_type do_thousands_sep () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.877.1 Detailed Description

template<typename _CharT, bool _Intl>class std::moneypunct< _CharT, _Intl >

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 934 of file locale_facets_nonio.h.

4.877.2 Member Typedef Documentation

4.877.2.1 template<typename _CharT , bool _Intl> typedef _CharT std::moneypunct< _CharT, _Intl >::char_type

Public typedefs.

Definition at line 940 of file locale_facets_nonio.h.

4.877.2.2 template<typename _CharT , bool _Intl> typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type

Public typedefs.

Definition at line 941 of file locale_facets_nonio.h.

4.877.3 Constructor & Destructor Documentation

4.877.3.1 template<typename _CharT , bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 963 of file locale_facets_nonio.h.

4.877.3.2 template<typename _CharT , bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct (__cache_type ∗ *__cache*, size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is an internal constructor.

Parameters

__cache	Cache for optimization.
__refs	Passed to the base facet class.

Definition at line 976 of file locale_facets_nonio.h.

4.877.3.3 template<typename _CharT , bool _Intl> std::moneypunct< _CharT, _Intl >::moneypunct (__c_locale *__cloc*, const char ∗ *__s*, size_t *__refs* = 0) [inline], [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__s	The name of a locale.
__refs	Passed to the base facet class.

Definition at line 991 of file locale_facets_nonio.h.

4.877.3.4 template<typename _CharT , bool _Intl> virtual std::moneypunct< _CharT, _Intl >::∼moneypunct ( ) [protected], [virtual]

Destructor.

4.877.4 Member Function Documentation

4.877.4.1 template<typename _CharT , bool _Intl> string_type std::moneypunct< _CharT, _Intl >::curr_symbol ( ) const [inline]

Return currency symbol string.

This function returns a string_type to use as a currency symbol. It does so by returning returning moneypunct<char_type>::do_curr_symbol().

Returns

string_type representing a currency symbol.

Definition at line 1061 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_curr_symbol().

4.877.4.2 template<typename _CharT , bool _Intl> char_type std::moneypunct< _CharT, _Intl >::decimal_point ( ) const [inline]

Return decimal point character.

This function returns a char_type to use as a decimal point. It does so by returning returning moneypunct<char_type>::do_decimal_point().

Returns

char_type representing a decimal point.

Definition at line 1005 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_decimal_point().

4.877.4.3 template<typename _CharT , bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol ( ) const [inline], [protected], [virtual]

Return currency symbol string.

This function returns a string_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See Also

curr_symbol() for details.

Returns

string_type representing a currency symbol.

Definition at line 1207 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::curr_symbol().

4.877.4.4 template<typename _CharT , bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point ( ) const [inline], [protected], [virtual]

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1169 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::decimal_point().

4.877.4.5 template<typename _CharT , bool _Intl> virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits ( ) const [inline], [protected], [virtual]

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See Also

frac_digits() for details.

Returns

Number of digits in amount fraction.

Definition at line 1247 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::frac_digits().

4.877.4.6 template<typename _CharT , bool _Intl> virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const [inline], [protected], [virtual]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See Also

grouping() for details.

Returns

String representing grouping specification.

Definition at line 1194 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::grouping().

4.877.4.7 template<typename _CharT , bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const [inline], [protected], [virtual]

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See Also

neg_format() for details.

Returns

Pattern for money values.

Definition at line 1275 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::neg_format().

4.877.4.8 template<typename _CharT , bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const [inline], [protected], [virtual]

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See Also

negative_sign() for details.

Returns

string_type representing a negative sign.

Definition at line 1233 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::negative_sign().

4.877.4.9 template<typename _CharT , bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const [inline], [protected], [virtual]

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See Also

pos_format() for details.

Returns

Pattern for money values.

Definition at line 1261 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::pos_format().

4.877.4.10 template<typename _CharT , bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const [inline], [protected], [virtual]

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See Also

positive_sign() for details.

Returns

string_type representing a positive sign.

Definition at line 1220 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::positive_sign().

4.877.4.11 template<typename _CharT , bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep ( ) const [inline], [protected], [virtual]

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1181 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::thousands_sep().

4.877.4.12 template<typename _CharT , bool _Intl> int std::moneypunct< _CharT, _Intl >::frac_digits ( ) const [inline]

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

Definition at line 1111 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_frac_digits().

4.877.4.13 template<typename _CharT , bool _Intl> string std::moneypunct< _CharT, _Intl >::grouping ( ) const `[inline]`

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

Definition at line 1048 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_grouping().

4.877.4.14 template<typename _CharT , bool _Intl> pattern std::moneypunct< _CharT, _Intl >::neg_format ( ) const `[inline]`

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1151 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_neg_format().

4.877.4.15 template<typename _CharT , bool _Intl> string_type std::moneypunct< _CharT, _Intl >::negative_sign ( ) const `[inline]`

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char_type>::do_negative_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1095 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_negative_sign().

4.877.4.16 template<typename _CharT , bool _Intl> pattern std::moneypunct< _CharT, _Intl >::pos_format ( ) const [inline]

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1147 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_pos_format().

4.877.4.17 template<typename _CharT , bool _Intl> string_type std::moneypunct< _CharT, _Intl >::positive_sign ( ) const [inline]

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char_type>::do_positive_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1078 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_positive_sign().

4.877.4.18 template<typename _CharT , bool _Intl> char_type std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const [inline]

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning moneypunct<char-_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

Definition at line 1018 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_thousands_sep().

4.877.5 Member Data Documentation

4.877.5.1 template<typename _CharT , bool _Intl> locale::id std::moneypunct< _CharT, _Intl >::id [static]

Numpunct facet id.

Definition at line 953 of file locale_facets_nonio.h.

4.877.5.2 template<typename _CharT , bool _Intl> const bool std::moneypunct< _CharT, _Intl >::intl [static]

This value is provided by the standard, but no reason for its existence.

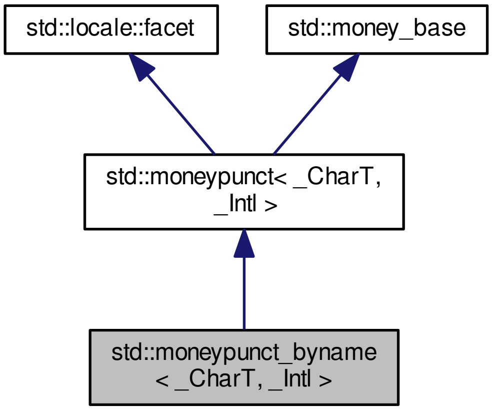
Definition at line 951 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

4.878 std::moneypunct_byname< _CharT, _Intl > Class Template Reference

Inheritance diagram for std::moneypunct_byname< _CharT, _Intl >:



Public Types

- enum { **_S_minus**, **_S_zero**, **_S_end** }
- typedef __moneypunct_cache
  < _CharT, _Intl > **__cache_type**
- typedef _CharT **char_type**
- enum **part** {
  none, **space**, **symbol**, **sign**,
  value }
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **moneypunct_byname** (const char ∗__s, size_t __refs=0)
- string_type curr_symbol () const
- char_type decimal_point () const
- int frac_digits () const
- string grouping () const
- string_type negative_sign () const
- string_type positive_sign () const
- char_type thousands_sep () const

- pattern pos_format () const
- pattern neg_format () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char ∗ **_S_atoms**
- static const pattern **_S_default_pattern**
- static locale::id id
- static const bool **intl**

Protected Member Functions

- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char ∗__name=0)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- template<>
  void **_M_initialize_moneypunct** (__c_locale, const char ∗)
- virtual string_type do_curr_symbol () const
- virtual char_type do_decimal_point () const
- virtual int do_frac_digits () const

- virtual string do_grouping () const
- virtual pattern do_neg_format () const
- virtual string_type do_negative_sign () const
- virtual pattern do_pos_format () const
- virtual string_type do_positive_sign () const
- virtual char_type do_thousands_sep () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.878.1 Detailed Description

template<typename _CharT, bool _Intl>class std::moneypunct_byname< _CharT, _Intl >

class moneypunct_byname [22.2.6.4].

Definition at line 1324 of file locale_facets_nonio.h.

4.878.2 Member Function Documentation

4.878.2.1 template<typename _CharT , bool _Intl> string_type std::moneypunct< _CharT, _Intl >::curr_symbol ( ) const [inline], [inherited]

Return currency symbol string.

This function returns a string_type to use as a currency symbol. It does so by returning returning moneypunct<char_type>::do_curr_symbol().

Returns

string_type representing a currency symbol.

Definition at line 1061 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_curr_symbol().

4.878.2.2 template<typename _CharT , bool _Intl> char_type std::moneypunct< _CharT, _Intl >::decimal_point ( ) const [inline], [inherited]

Return decimal point character.

This function returns a char_type to use as a decimal point. It does so by returning returning moneypunct<char_type>::do_decimal_point().

Returns

char_type representing a decimal point.

Definition at line 1005 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_decimal_point().

4.878.2.3 template<typename _CharT , bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol ( ) const [inline], [protected], [virtual], [inherited]

Return currency symbol string.

This function returns a string_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See Also

curr_symbol() for details.

Returns

string_type representing a currency symbol.

Definition at line 1207 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::curr_symbol().

4.878.2.4 template<typename _CharT , bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point ( ) const [inline], [protected], [virtual], [inherited]

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1169 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::decimal_point().

4.878.2.5 template<typename _CharT , bool _Intl> virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits ( ) const [inline], [protected], [virtual], [inherited]

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See Also

frac_digits() for details.

Returns

Number of digits in amount fraction.

Definition at line 1247 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::frac_digits().

4.878.2.6 template<typename _CharT , bool _Intl> virtual string std::moneypunct< _CharT, _Intl >::do_grouping ( ) const [inline], [protected], [virtual], [inherited]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See Also

grouping() for details.

Returns

String representing grouping specification.

Definition at line 1194 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::grouping().

4.878.2.7 template<typename _CharT , bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format ( ) const [inline], [protected], [virtual], [inherited]

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See Also

neg_format() for details.

Returns

Pattern for money values.

Definition at line 1275 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::neg_format().

4.878.2.8 template<typename _CharT , bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign ( ) const [inline], [protected], [virtual], [inherited]

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See Also

negative_sign() for details.

Returns

string_type representing a negative sign.

Definition at line 1233 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::negative_sign().

4.878.2.9 template<typename _CharT , bool _Intl> virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format ( ) const [inline], [protected], [virtual], [inherited]

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See Also

pos_format() for details.

Returns

Pattern for money values.

Definition at line 1261 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::pos_format().

4.878.2.10 template<typename _CharT , bool _Intl> virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign ( ) const [inline], [protected], [virtual], [inherited]

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See Also

positive_sign() for details.

Returns

string_type representing a positive sign.

Definition at line 1220 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::positive_sign().

4.878.2.11 template<typename _CharT , bool _Intl> virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep ( ) const [inline], [protected], [virtual], [inherited]

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1181 of file locale_facets_nonio.h.

Referenced by std::moneypunct< _CharT, _Intl >::thousands_sep().

4.878.2.12 template<typename _CharT , bool _Intl> int std::moneypunct< _CharT, _Intl >::frac_digits ( ) const [inline], [inherited]

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

Definition at line 1111 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_frac_digits().

4.878.2.13 template<typename _CharT , bool _Intl> string std::moneypunct< _CharT, _Intl >::grouping ( ) const [inline], [inherited]

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

Definition at line 1048 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_grouping().

4.878.2.14 template<typename _CharT , bool _Intl> pattern std::moneypunct< _CharT, _Intl >::neg_format ( ) const [inline], [inherited]

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1151 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_neg_format().

4.878.2.15 template<typename _CharT , bool _Intl> string_type std::moneypunct< _CharT, _Intl >::negative_sign ( ) const [inline], [inherited]

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char_type>::do_negative_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

Definition at line 1095 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_negative_sign().

4.878.2.16 template<typename _CharT , bool _Intl> pattern std::moneypunct< _CharT, _Intl >::pos_format ( ) const [inline], [inherited]

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

Definition at line 1147 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_pos_format().

4.878.2.17 template<typename _CharT , bool _Intl> string_type std::moneypunct< _CharT, _Intl >::positive_sign ( ) const [inline], [inherited]

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char_type>::do_positive_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

Definition at line 1078 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_positive_sign().

4.878.2.18 template<typename _CharT , bool _Intl> **char_type std::moneypunct< _CharT, _Intl >::thousands_sep ( ) const** `[inline], [inherited]`

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning moneypunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

Definition at line 1018 of file locale_facets_nonio.h.

References std::moneypunct< _CharT, _Intl >::do_thousands_sep().

4.878.3 Member Data Documentation

4.878.3.1 template<typename _CharT , bool _Intl> **locale::id std::moneypunct< _CharT, _Intl >::id** `[static], [inherited]`

Numpunct facet id.

Definition at line 953 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

4.879 std::move_iterator< _Iterator > Class Template Reference

Public Types

- typedef
  __traits_type::difference_type **difference_type**
- typedef
  __traits_type::iterator_category **iterator_category**
- typedef _Iterator **iterator_type**
- typedef _Iterator **pointer**
- typedef value_type && **reference**
- typedef __traits_type::value_type **value_type**

Public Member Functions

- **move_iterator** (iterator_type __i)
- template<typename _Iter >
  move_iterator (const move_iterator< _Iter > &__i)

- iterator_type **base** () const
- reference **operator*** () const
- move_iterator **operator+** (difference_type __n) const
- move_iterator & **operator++** ()
- move_iterator **operator++** (int)
- move_iterator & **operator+=** (difference_type __n)
- move_iterator **operator-** (difference_type __n) const
- move_iterator & **operator--** ()
- move_iterator **operator--** (int)
- move_iterator & **operator-=** (difference_type __n)
- pointer **operator->** () const
- reference **operator[ ]** (difference_type __n) const

Protected Types

- typedef iterator_traits
  < _Iterator > **__traits_type**

Protected Attributes

- _Iterator **_M_current**

4.879.1 Detailed Description

template<typename _Iterator>class std::move_iterator< _Iterator >

Class template move_iterator is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

Definition at line 930 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- stl_iterator.h

4.880 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef
  _Pair_alloc_type::const_pointer **const_pointer**
- typedef
  _Pair_alloc_type::const_reference **const_reference**
- typedef
  _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**

- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Pair_alloc_type::pointer **pointer**
- typedef _Pair_alloc_type::reference **reference**
- typedef _Rep_type::reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef std::pair< const _Key,
  _Tp > **value_type**

Public Member Functions

- multimap ()
- multimap (const _Compare &__comp, const allocator_type &__a=allocator_type())
- multimap (const multimap &__x)
- multimap (multimap &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- multimap (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  multimap (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
  multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- iterator begin () noexcept
- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- size_type count (const key_type &__x) const
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > equal_range (const key_type &__x) const
- iterator erase (const_iterator __position)
- iterator **erase** (iterator __position)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const
- allocator_type get_allocator () const noexcept
- iterator insert (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__x)
- iterator insert (const_iterator __position, const value_type &__x)

- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_compare key_comp () const
- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const
- size_type max_size () const noexcept
- multimap & operator= (const multimap &__x)
- multimap & operator= (multimap &&__x)
- multimap & operator= (initializer_list< value_type > __l)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (multimap &__x)
- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const
- value_compare value_comp () const

Friends

- template<typename _K1 , typename _T1 , typename _C1 , typename _A1 >
  bool **operator<** (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)
- template<typename _K1 , typename _T1 , typename _C1 , typename _A1 >
  bool **operator==** (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)

4.880.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>class std::multimap< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>.

Meets the requirements of a `container`, a `reversible container`, and an `associative container` (using equivalent keys). For a `multimap<Key,T>` the key_type is Key, the mapped_type is T, and the value_type is std::pair<const Key,T>.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (∗_unique versus ∗_equal, same as the standard).

Definition at line 95 of file stl_multimap.h.

4.880.2 Constructor & Destructor Documentation

4.880.2.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> **std::multimap**< _Key, _Tp, _Compare, _Alloc >::**multimap** ( ) `[inline]`

Default constructor creates no elements.

Definition at line 157 of file stl_multimap.h.

4.880.2.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> **std::multimap**< _Key, _Tp, _Compare, _Alloc >::**multimap** (const _Compare & *__comp*, const allocator_type & *__a* = `allocator_type()`) `[inline]`, `[explicit]`

Creates a multimap with no elements.

Parameters

__comp	A comparison object.
__a	An allocator object.

Definition at line 166 of file stl_multimap.h.

4.880.2.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> **std::multimap**< _Key, _Tp, _Compare, _Alloc >::**multimap** (const **multimap**< _Key, _Tp, _Compare, _Alloc > & *__x*) `[inline]`

Multimap copy constructor.

Parameters

__x	A multimap of identical element and allocator types.

The newly-created multimap uses a copy of the allocation object used by *__x*.

Definition at line 177 of file stl_multimap.h.

4.880.2.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> **std::multimap**< _Key, _Tp, _Compare, _Alloc >::**multimap** (**multimap**< _Key, _Tp, _Compare, _Alloc > && *__x*) `[inline]`, `[noexcept]`

Multimap move constructor.

Parameters

__x	A multimap of identical element and allocator types.

The newly-created multimap contains the exact contents of *__x*. The contents of *__x* are a valid, but unspecified multimap.

Definition at line 188 of file stl_multimap.h.

4.880.2.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> **std::multimap**< _Key, _Tp, _Compare, _Alloc >::**multimap** (**initializer_list**< **value_type** > *__l*, const _Compare & *__comp* = `_Compare()`, const allocator_type & *__a* = `allocator_type()`) `[inline]`

Builds a multimap from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a multimap consisting of copies of the elements from the initializer_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

Definition at line 202 of file stl_multimap.h.

4.880.2.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (_InputIterator *__first*, _InputIterator *__last*) `[inline]`

Builds a multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a multimap consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 219 of file stl_multimap.h.

4.880.2.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (_InputIterator *__first*, _InputIterator *__last*, const _Compare & *__comp*, const allocator_type & *__a* = `allocator_type()`) `[inline]`

Builds a multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a multimap consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 235 of file stl_multimap.h.

4.880.3 Member Function Documentation

4.880.3.1 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ( ) `[inline]`, `[noexcept]`

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 314 of file stl_multimap.h.

4.880.3.2 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 323 of file stl_multimap.h.

4.880.3.3 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 387 of file stl_multimap.h.

4.880.3.4 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 396 of file stl_multimap.h.

4.880.3.5 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::multimap< _Key, _Tp, _Compare, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 696 of file stl_multimap.h.

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=().

4.880.3.6 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::count (const key_type & __x) const [inline]

Finds the number of elements with given key.

Parameters

__x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

Definition at line 753 of file stl_multimap.h.

4.880.3.7 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 405 of file stl_multimap.h.

4.880.3.8 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 414 of file stl_multimap.h.

4.880.3.9 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (_Args &&... *__args*) [inline]

Build and insert a std::pair into the multimap.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 454 of file stl_multimap.h.

4.880.3.10 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace_hint (const_iterator *__pos*, _Args &&... *__args*) [inline]

Builds and inserts a std::pair into the multimap.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html`

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 481 of file stl_multimap.h.

4.880.3.11 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> bool std::multimap< _Key, _Tp, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the multimap is empty.

Definition at line 421 of file stl_multimap.h.

4.880.3.12 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 332 of file stl_multimap.h.

4.880.3.13 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 341 of file stl_multimap.h.

4.880.3.14 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, iterator> std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (const key_type & __x) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 820 of file stl_multimap.h.

4.880.3.15 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator> std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (const key_type & __x) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 837 of file stl_multimap.h.

4.880.3.16 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (const_iterator __position) [inline]

Erases an element from a multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 595 of file stl_multimap.h.

4.880.3.17 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (const key_type & __x) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 630 of file stl_multimap.h.

4.880.3.18 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (const_iterator *__first,* const_iterator *__last*) [inline]

Erases a [first,last) range of elements from a multimap.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased .

Returns

The iterator __*last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 651 of file stl_multimap.h.

4.880.3.19 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (const key_type & *__x*) [inline]

Tries to locate an element in a multimap.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 729 of file stl_multimap.h.

4.880.3.20 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (const key_type & *__x*) const [inline]

Tries to locate an element in a multimap.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 744 of file stl_multimap.h.

4.880.3.21 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator ( ) const `[inline]`, `[noexcept]`

Get a copy of the memory allocation object.

Definition at line 304 of file stl_multimap.h.

4.880.3.22 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (const value_type & __x) `[inline]`

Inserts a std::pair into the multimap.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 501 of file stl_multimap.h.

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=().

4.880.3.23 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (const_iterator __position, const value_type & __x) `[inline]`

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 535 of file stl_multimap.h.

4.880.3.24 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void **std::multimap**< _Key, _Tp, _Compare, _Alloc >::insert (_InputIterator *__first*, _InputIterator *__last*) [inline]

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 562 of file stl_multimap.h.

4.880.3.25 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void **std::multimap**< _Key, _Tp, _Compare, _Alloc >::insert (**initializer_list< value_type >** *__l*) [inline]

Attempts to insert a list of std::pairs into the multimap.

Parameters

__l	A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

Definition at line 574 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

4.880.3.26 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> key_compare **std::multimap**< _Key, _Tp, _Compare, _Alloc >::key_comp ( ) const [inline]

Returns the key comparison object out of which the multimap was constructed.

Definition at line 705 of file stl_multimap.h.

4.880.3.27 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator **std::multimap**< _Key, _Tp, _Compare, _Alloc >::lower_bound (const key_type & *__x*) [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 768 of file stl_multimap.h.

4.880.3.28 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (const key_type & __x) const [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 783 of file stl_multimap.h.

4.880.3.29 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the multimap.

Definition at line 431 of file stl_multimap.h.

4.880.3.30 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> multimap& std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (const multimap< _Key, _Tp, _Compare, _Alloc > & __x) [inline]

Multimap assignment operator.

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Parameters

__x	A multimap of identical element and allocator types.

All the elements of *__x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 258 of file stl_multimap.h.

4.880.3.31 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> multimap& std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (multimap< _Key, _Tp, _Compare, _Alloc > && __x) [inline]

Multimap move assignment operator.

Parameters

__x	A multimap of identical element and allocator types.

The contents of *__x* are moved into this multimap (without copying). *__x* is a valid, but unspecified multimap.

Definition at line 273 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::clear(), and std::multimap< _Key, _Tp, _Compare, _Alloc >::swap().

4.880.3.32 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> multimap& std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Multimap list assignment operator.

Parameters

__l	An initializer_list.

This function fills a multimap with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 294 of file stl_multimap.h.

References std::multimap< _Key, _Tp, _Compare, _Alloc >::clear(), and std::multimap< _Key, _Tp, _Compare, _Alloc >::insert().

4.880.3.33 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 350 of file stl_multimap.h.

4.880.3.34 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 359 of file stl_multimap.h.

4.880.3.35 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 368 of file stl_multimap.h.

4.880.3.36 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 377 of file stl_multimap.h.

4.880.3.37 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the multimap.

Definition at line 426 of file stl_multimap.h.

4.880.3.38 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (multimap< _Key, _Tp, _Compare, _Alloc > & __x) [inline]

Swaps data with another multimap.

Parameters

__x	A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 686 of file stl_multimap.h.

Referenced by std::multimap< _Key, _Tp, _Compare, _Alloc >::operator=(), and std::swap().

4.880.3.39 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (const key_type & __x) [inline]

Finds the end of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 793 of file stl_multimap.h.

4.880.3.40 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator **std::multimap**< _Key, _Tp, _Compare, _Alloc >::upper_bound (const key_type & __x) const `[inline]`

Finds the end of a subsequence matching given key.

Parameters

__x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 803 of file stl_multimap.h.

4.880.3.41 template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >> value_compare **std::multimap**< _Key, _Tp, _Compare, _Alloc >::value_comp ( ) const `[inline]`

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

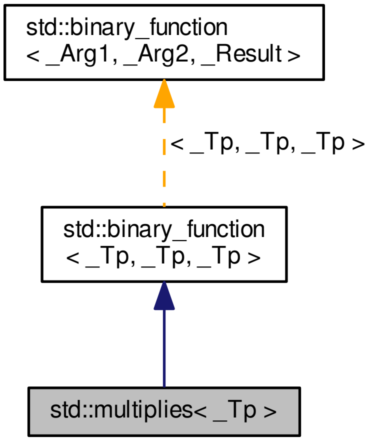
Definition at line 713 of file stl_multimap.h.

The documentation for this class was generated from the following file:

- stl_multimap.h

4.881 std::multiplies< _Tp > Struct Template Reference

Inheritance diagram for std::multiplies< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

4.881.1 Detailed Description

template<typename _Tp>struct std::multiplies< _Tp >

One of the math functors.

Definition at line 158 of file stl_function.h.

4.881.2 Member Typedef Documentation

4.881.2.1 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.881.2.2 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.881.2.3 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.882 std::multiset< _Key, _Compare, _Alloc > Class Template Reference

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Key_alloc_type::const_pointer **const_pointer**
- typedef _Key_alloc_type::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**

- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::const_iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Key_alloc_type::pointer **pointer**
- typedef _Key_alloc_type::reference **reference**
- typedef
  _Rep_type::const_reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- multiset ()
- multiset (const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  multiset (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- multiset (const multiset &__x)
- multiset (multiset &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- multiset (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- iterator begin () const noexcept
- iterator cbegin () const noexcept
- iterator cend () const noexcept
- void clear () noexcept
- size_type count (const key_type &__x) const
- reverse_iterator crbegin () const noexcept
- reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- iterator end () const noexcept
- iterator erase (const_iterator __position)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- iterator insert (const value_type &__x)
- iterator **insert** (value_type &&__x)
- iterator insert (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_compare key_comp () const
- size_type max_size () const noexcept

- multiset & operator= (const multiset &__x)
- multiset & operator= (multiset &&__x)
- multiset & operator= (initializer_list< value_type > __l)
- reverse_iterator rbegin () const noexcept
- reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (multiset &__x)
- value_compare value_comp () const

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const

- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > equal_range (const key_type &__x) const

Friends

- template<typename _K1 , typename _C1 , typename _A1 >
  bool **operator<** (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)
- template<typename _K1 , typename _C1 , typename _A1 >
  bool **operator==** (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)

4.882.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>class std::multiset< _Key, _Compare, _Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time.

Template Parameters

_Key	Type of key objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a `container`, a `reversible container`, and an `associative container` (using equivalent keys). For a `multiset<Key>` the key_type and value_type are Key.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (∗_unique versus ∗_equal, same as the standard).

Definition at line 92 of file stl_multiset.h.

4.882.2 Constructor & Destructor Documentation

4.882.2.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset< _Key, _Compare, _Alloc >::multiset ( ) [inline]

Default constructor creates no elements.

Definition at line 137 of file stl_multiset.h.

4.882.2.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset< _Key, _Compare, _Alloc >::multiset (const _Compare & *__comp*, const allocator_type & *__a* = allocator_type()) [inline], [explicit]

Creates a multiset with no elements.

Parameters

__comp	Comparator to use.
__a	An allocator object.

Definition at line 146 of file stl_multiset.h.

4.882.2.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset (_InputIterator *__first*, _InputIterator *__last*) [inline]

Builds a multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 160 of file stl_multiset.h.

4.882.2.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > std::multiset< _Key, _Compare, _Alloc >::multiset (_InputIterator *__first*, _InputIterator *__last*, const _Compare & *__comp*, const allocator_type & *__a* = allocator_type()) [inline]

Builds a multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a multiset consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 176 of file stl_multiset.h.

4.882.2.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset< _Key, _Compare, _Alloc >::multiset (const multiset< _Key, _Compare, _Alloc > & __x) [inline]

Multiset copy constructor.

Parameters

__x	A multiset of identical element and allocator types.

The newly-created multiset uses a copy of the allocation object used by *__x*.

Definition at line 189 of file stl_multiset.h.

4.882.2.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset< _Key, _Compare, _Alloc >::multiset (multiset< _Key, _Compare, _Alloc > && __x) [inline], [noexcept]

Multiset move constructor.

Parameters

__x	A multiset of identical element and allocator types.

The newly-created multiset contains the exact contents of *__x*. The contents of *__x* are a valid, but unspecified multiset.

Definition at line 200 of file stl_multiset.h.

4.882.2.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::multiset< _Key, _Compare, _Alloc >::multiset (initializer_list< value_type > __l, const _Compare & __comp = _Compare(), const allocator_type & __a = allocator_type()) [inline]

Builds a multiset from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

Definition at line 214 of file stl_multiset.h.

4.882.3 Member Function Documentation

4.882.3.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 295 of file stl_multiset.h.

4.882.3.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 332 of file stl_multiset.h.

4.882.3.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 341 of file stl_multiset.h.

4.882.3.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::multiset< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 614 of file stl_multiset.h.

Referenced by std::multiset< _Key, _Compare, _Alloc >::operator=().

4.882.3.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::multiset< _Key, _Compare, _Alloc >::count (const key_type & __x) const [inline]

Finds the number of elements with given key.

Parameters

__x	Key of elements to be located.

Returns

Number of elements with specified key.

Definition at line 625 of file stl_multiset.h.

4.882.3.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 350 of file stl_multiset.h.

4.882.3.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 359 of file stl_multiset.h.

4.882.3.8 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename... _Args> iterator std::multiset< _Key, _Compare, _Alloc >::emplace (_Args &&... __args) [inline]

Builds and inserts an element into the multiset.

Parameters

__args	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 409 of file stl_multiset.h.

4.882.3.9 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename... _Args> iterator std::multiset< _Key, _Compare, _Alloc >::emplace_hint (const_iterator __pos, _Args &&... __args) [inline]

Builds and inserts an element into the multiset.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 435 of file stl_multiset.h.

4.882.3.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool std::multiset< _Key, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the set is empty.

Definition at line 365 of file stl_multiset.h.

4.882.3.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 304 of file stl_multiset.h.

4.882.3.12 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::pair<iterator, iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (const key_type & __x) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 705 of file stl_multiset.h.

4.882.3.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::pair<const_iterator, const_iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (const key_type & __x) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 709 of file stl_multiset.h.

4.882.3.14 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::erase (const_iterator __position) [inline]

Erases an element from a multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 536 of file stl_multiset.h.

4.882.3.15 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::multiset< _Key, _Compare, _Alloc >::erase (const key_type & __x) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 566 of file stl_multiset.h.

4.882.3.16 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::erase (const_iterator __first, const_iterator __last) [inline]

Erases a [first,last) range of elements from a multiset.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 587 of file stl_multiset.h.

4.882.3.17 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator **std::multiset**< _Key, _Compare, _Alloc >::find (const key_type & *__x*) [inline]

Tries to locate an element in a set.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 643 of file stl_multiset.h.

4.882.3.18 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> const_iterator **std::multiset**< _Key, _Compare, _Alloc >::find (const key_type & *__x*) const [inline]

Tries to locate an element in a set.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 647 of file stl_multiset.h.

4.882.3.19 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> allocator_type **std::multiset**< _Key, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Returns the memory allocation object.

Definition at line 286 of file stl_multiset.h.

4.882.3.20 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator **std::multiset**< _Key, _Compare, _Alloc >::insert (const value_type & *__x*) [inline]

Inserts an element into the multiset.

Parameters

__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 454 of file stl_multiset.h.

Referenced by std::multiset< _Key, _Compare, _Alloc >::operator=().

4.882.3.21 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator **std::multiset**< _Key, _Compare, _Alloc >::insert (const_iterator *__position,* const value_type & *__x*) [inline]

Inserts an element into the multiset.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 484 of file stl_multiset.h.

4.882.3.22 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > void **std::multiset**< _Key, _Compare, _Alloc >::insert (_InputIterator *__first,* _InputIterator *__last*) [inline]

A template function that tries to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 503 of file stl_multiset.h.

4.882.3.23 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void **std::multiset**< _Key, _Compare, _Alloc >::insert (**initializer_list**< value_type > *__l*) [inline]

Attempts to insert a list of elements into the multiset.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 515 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::insert().

Referenced by std::multiset< _Key, _Compare, _Alloc >::insert().

4.882.3.24 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp ( ) const [inline]

Returns the comparison object.

Definition at line 278 of file stl_multiset.h.

4.882.3.25 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (const key_type & __x) [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 664 of file stl_multiset.h.

4.882.3.26 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (const key_type & __x) const [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 668 of file stl_multiset.h.

4.882.3.27 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::multiset< _Key, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the set.

Definition at line 375 of file stl_multiset.h.

4.882.3.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (const multiset< _Key, _Compare, _Alloc > & __x) [inline]

Multiset assignment operator.

Parameters

__x	A multiset of identical element and allocator types.

All the elements of *__x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 229 of file stl_multiset.h.

4.882.3.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (multiset< _Key, _Compare, _Alloc > && __x) [inline]

Multiset move assignment operator.

Parameters

__x	A multiset of identical element and allocator types.

The contents of *__x* are moved into this multiset (without copying). *__x* is a valid, but unspecified multiset.

Definition at line 245 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::clear(), and std::multiset< _Key, _Compare, _Alloc >::swap().

4.882.3.30 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Multiset list assignment operator.

Parameters

__l	An initializer_list.

This function fills a multiset with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 266 of file stl_multiset.h.

References std::multiset< _Key, _Compare, _Alloc >::clear(), and std::multiset< _Key, _Compare, _Alloc >::insert().

4.882.3.31 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 313 of file stl_multiset.h.

4.882.3.32 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend ( ) const `[inline]`, `[noexcept]`

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 322 of file stl_multiset.h.

4.882.3.33 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::multiset< _Key, _Compare, _Alloc >::size ( ) const `[inline]`, `[noexcept]`

Returns the size of the set.

Definition at line 370 of file stl_multiset.h.

4.882.3.34 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::multiset< _Key, _Compare, _Alloc >::swap (multiset< _Key, _Compare, _Alloc > & __x) `[inline]`

Swaps data with another multiset.

Parameters

__x	A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

Definition at line 390 of file stl_multiset.h.

Referenced by std::multiset< _Key, _Compare, _Alloc >::operator=(), and std::swap().

4.882.3.35 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (const key_type & __x) `[inline]`

Finds the end of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 680 of file stl_multiset.h.

4.882.3.36 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (const key_type & __x) const `[inline]`

Finds the end of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 684 of file stl_multiset.h.

4.882.3.37 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> value_compare **std::multiset**< _Key, _Compare, _Alloc >::value_comp ( ) const [inline]

Returns the comparison object.

Definition at line 282 of file stl_multiset.h.

The documentation for this class was generated from the following file:

- stl_multiset.h

4.883 std::mutex Class Reference

Inherits std::__mutex_base.

Public Types

- typedef __native_type ∗ **native_handle_type**

Public Member Functions

- **mutex** (const mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** ()
- mutex & **operator=** (const mutex &)=delete
- bool **try_lock** () noexcept
- void **unlock** ()

Private Types

- typedef __gthread_mutex_t **__native_type**

Private Attributes

- __native_type **_M_mutex**

4.883.1 Detailed Description

mutex

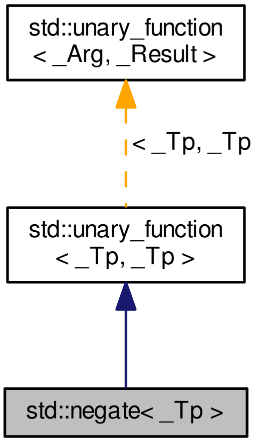
Definition at line 116 of file mutex.

The documentation for this class was generated from the following file:

- mutex

4.884 std::negate< _Tp > Struct Template Reference

Inheritance diagram for std::negate< _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Tp result_type

Public Member Functions

- _Tp **operator()** (const _Tp &__x) const

4.884.1 Detailed Description

template<typename _Tp>struct std::negate< _Tp >

One of the math functors.

Definition at line 185 of file stl_function.h.

4.884.2 Member Typedef Documentation

4.884.2.1 typedef _Tp std::unary_function< _Tp , _Tp >::argument_type [inherited]

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.884.2.2 typedef _Tp **std::unary_function**< _Tp , _Tp >**::result_type** `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.885 std::negative_binomial_distribution< _IntType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **negative_binomial_distribution** (_IntType __k=1, double __p=0.5)
- **negative_binomial_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _IntType k () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::negative_binomial_distribution< _IntType1 > &__x)

- bool operator== (const negative_binomial_distribution &__d1, const negative_binomial_distribution &__d2)
- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::negative_binomial_distribution< _IntType1 > &__x)

4.885.1 Detailed Description

template<typename _IntType = int>class std::negative_binomial_distribution< _IntType >

A negative_binomial_distribution random number distribution.

The formula for the negative binomial probability mass function is $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Definition at line 4207 of file random.h.

4.885.2 Member Typedef Documentation

4.885.2.1 template<typename _IntType = int> typedef _IntType std::negative_binomial_distribution< _IntType >::result_type

The type of the range of the distribution.

Definition at line 4210 of file random.h.

4.885.3 Member Function Documentation

4.885.3.1 template<typename _IntType = int> _IntType std::negative_binomial_distribution< _IntType >::k ( ) const [inline]

Return the k parameter of the distribution.

Definition at line 4265 of file random.h.

4.885.3.2 template<typename _IntType = int> result_type std::negative_binomial_distribution< _IntType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 4301 of file random.h.

References std::numeric_limits< _Tp >::max().

4.885.3.3 template<typename _IntType = int> result_type std::negative_binomial_distribution< _IntType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4294 of file random.h.

4.885.3.4 template<typename _IntType > template<typename _UniformRandomNumberGenerator > negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*)

Generating functions.

Definition at line 1278 of file bits/random.tcc.

4.885.3.5 template<typename _IntType = int> double std::negative_binomial_distribution< _IntType >::p ( ) const [inline]

Return the p parameter of the distribution.

Definition at line 4272 of file random.h.

4.885.3.6 template<typename _IntType = int> param_type std::negative_binomial_distribution< _IntType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 4279 of file random.h.

4.885.3.7 template<typename _IntType = int> void std::negative_binomial_distribution< _IntType >::param (const param_type & __param) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 4287 of file random.h.

4.885.3.8 template<typename _IntType = int> void std::negative_binomial_distribution< _IntType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 4258 of file random.h.

References std::gamma_distribution< _RealType >::reset().

4.885.4 Friends And Related Function Documentation

4.885.4.1 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::negative_binomial_distribution< _IntType1 > & __x) [friend]

Inserts a negative_binomial_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A negative_binomial_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.885.4.2 template<typename _IntType = int> bool operator== (const negative_binomial_distribution< _IntType > & *__d1*, const negative_binomial_distribution< _IntType > & *__d2*) [friend]

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4350 of file random.h.

4.885.4.3 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & *__is*, std::negative_binomial_distribution< _IntType1 > & *__x*) [friend]

Extracts a negative_binomial_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A negative_binomial_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.886 std::negative_binomial_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef negative_binomial_distribution < _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __k=1, double __p=0.5)
- _IntType **k** () const
- double **p** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.886.1 Detailed Description

template<typename _IntType = int>struct std::negative_binomial_distribution< _IntType >::param_type

Parameter type.

Definition at line 4216 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.887 std::nested_exception Class Reference

Inherited by std::_Nested_exception< _Except >.

Public Member Functions

- **nested_exception** (const nested_exception &)=default
- exception_ptr **nested_ptr** () const
- nested_exception & **operator=** (const nested_exception &)=default
- void **rethrow_nested** () const __attribute__((__noreturn__))

4.887.1 Detailed Description

Exception class with exception_ptr data member.

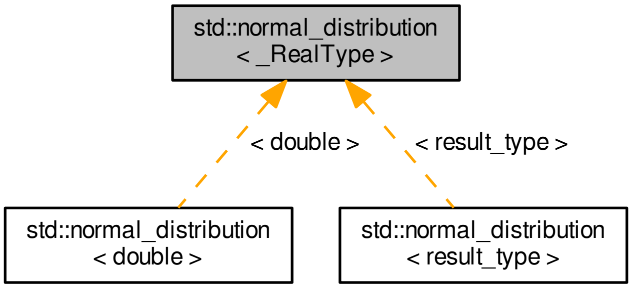
Definition at line 55 of file nested_exception.h.

The documentation for this class was generated from the following file:

- nested_exception.h

4.888 std::normal_distribution< _RealType > Class Template Reference

Inheritance diagram for std::normal_distribution< _RealType >:



Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- normal_distribution (result_type __mean=result_type(0), result_type __stddev=result_type(1))
- **normal_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- _RealType mean () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _RealType stddev () const

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::normal_distribution< _RealType1 > &__x)
- template<typename _RealType1 >
  bool operator== (const std::normal_distribution< _RealType1 > &__d1, const std::normal_distribution< _RealType1 > &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::normal_distribution< _RealType1 > &__x)

4.888.1 Detailed Description

template<typename _RealType = double>class std::normal_distribution< _RealType >

A normal continuous distribution for random numbers.

The formula for the normal probability density function is

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}}$$

Definition at line 2085 of file random.h.

4.888.2 Member Typedef Documentation

4.888.2.1 template<typename _RealType = double> typedef _RealType std::normal_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 2088 of file random.h.

4.888.3 Constructor & Destructor Documentation

4.888.3.1 template<typename _RealType = double> std::normal_distribution< _RealType >::normal_distribution (result_type *__mean* = result_type `(0)`, result_type *__stddev* = result_type `(1)`) `[inline]`, `[explicit]`

Constructs a normal distribution with parameters $mean$ and standard deviation.

Definition at line 2130 of file random.h.

4.888.4 Member Function Documentation

4.888.4.1 template<typename _RealType = double> result_type std::normal_distribution< _RealType >::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 2187 of file random.h.

4.888.4.2 template<typename _RealType = double> _RealType std::normal_distribution< _RealType >::mean ( ) const `[inline]`

Returns the mean of the distribution.

Definition at line 2151 of file random.h.

4.888.4.3 template<typename _RealType = double> result_type std::normal_distribution< _RealType >::min ( ) const `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 2180 of file random.h.

4.888.4.4 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::normal_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 2195 of file random.h.

Referenced by std::normal_distribution< result_type >::operator()().

4.888.4.5 template<typename _RealType > template<typename _UniformRandomNumberGenerator > normal_distribution< _RealType >::result_type std::normal_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*, const param_type & *__param*)

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1933 of file bits/random.tcc.

References std::log(), and std::sqrt().

4.888.4.6 template<typename _RealType = double> param_type std::normal_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 2165 of file random.h.

4.888.4.7 template<typename _RealType = double> void std::normal_distribution< _RealType >::param (const param_type & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 2173 of file random.h.

4.888.4.8 template<typename _RealType = double> void std::normal_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 2144 of file random.h.

Referenced by std::lognormal_distribution< _RealType >::reset(), std::gamma_distribution< result_type >::reset(), std::student_t_distribution< _RealType >::reset(), std::binomial_distribution< _IntType >::reset(), and std::poisson_distribution< _IntType >::reset().

4.888.4.9 template<typename _RealType = double> _RealType std::normal_distribution< _RealType >::stddev ( ) const [inline]

Returns the standard deviation of the distribution.

Definition at line 2158 of file random.h.

4.888.5 Friends And Related Function Documentation

4.888.5.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::normal_distribution< _RealType1 > & __x) [friend]

Inserts a normal_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A normal_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.888.5.2 template<typename _RealType = double> template<typename _RealType1 > bool operator== (const std::normal_distribution< _RealType1 > & __d1, const std::normal_distribution< _RealType1 > & __d2) [friend]

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

4.888.5.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & __is, std::normal_distribution< _RealType1 > & __x) [friend]

Extracts a normal_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A normal_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.889 std::normal_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef normal_distribution < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __mean=_RealType(0), _RealType __stddev=_RealType(1))
- _RealType **mean** () const
- _RealType **stddev** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.889.1 Detailed Description

template<typename _RealType = double>struct std::normal_distribution< _RealType >::param_type

Parameter type.

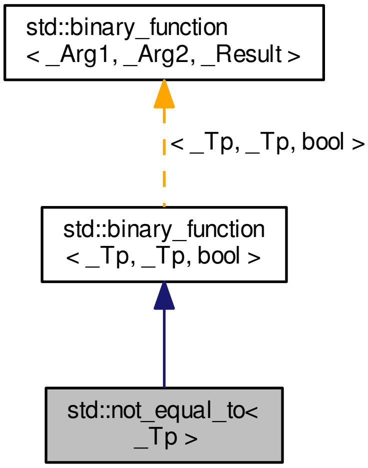
Definition at line 2094 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.890 std::not_equal_to< _Tp > Struct Template Reference

Inheritance diagram for std::not_equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__x, const _Tp &__y) const

4.890.1 Detailed Description

template<typename _Tp>struct std::not_equal_to< _Tp >

One of the comparison functors.

Definition at line 213 of file stl_function.h.

4.890.2 Member Typedef Documentation

4.890.2.1 typedef _Tp **std::binary_function**< _Tp , _Tp , bool >**::first_argument_type** [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.890.2.2 typedef bool **std::binary_function**< _Tp , _Tp , bool >**::result_type** [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.890.2.3 typedef _Tp **std::binary_function**< _Tp , _Tp , bool >**::second_argument_type** [inherited]

`second_argument_type` is the type of the second argument

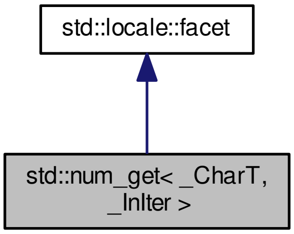
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.891 std::num_get< _CharT, _InIter > Class Template Reference

Inheritance diagram for std::num_get< _CharT, _InIter >:



Public Types

- typedef _CharT char_type

- typedef _InIter iter_type

Public Member Functions

- num_get (size_t __refs=0)
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, bool &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, void ∗&__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, float &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, double &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__v) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼num_get ()
- iter_type **_M_extract_float** (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const
- template<typename _ValueT >
  iter_type **_M_extract_int** (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const
- template<typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, int >::__type **_M_find** (const _CharT2 ∗, size_t __len, _CharT2 __c) const
- template<typename _CharT2 >
  __gnu_cxx::__enable_if
  <!__is_char< _CharT2 >
  ::__value, int >::__type **_M_find** (const _CharT2 ∗__zero, size_t __len, _CharT2 __c) const

- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const

- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void ∗&) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.891.1 Detailed Description

template<typename _CharT, typename _InIter>class std::num_get< _CharT, _InIter >

Primary class template num_get.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

The num_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_get facet.

Definition at line 1915 of file locale_facets.h.

4.891.2 Member Typedef Documentation

4.891.2.1 template<typename _CharT , typename _InIter > typedef _CharT std::num_get< _CharT, _InIter >::char_type

Public typedefs.

Definition at line 1921 of file locale_facets.h.

4.891.2.2 template<typename _CharT , typename _InIter > typedef _InIter std::num_get< _CharT, _InIter >::iter_type

Public typedefs.

Definition at line 1922 of file locale_facets.h.

4.891.3 Constructor & Destructor Documentation

4.891.3.1 template<typename _CharT , typename _InIter > std::num_get< _CharT, _InIter >::num_get (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 1936 of file locale_facets.h.

4.891.3.2 template<typename _CharT , typename _InIter > virtual std::num_get< _CharT, _InIter >::∼num_get ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 2108 of file locale_facets.h.

4.891.4 Member Function Documentation

4.891.4.1 template<typename _CharT , typename _InIter > _InIter std::num_get< _CharT, _InIter >::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v) const [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 590 of file locale_facets.tcc.

References std::ios_base::_M_getloc(), std::ios_base::boolalpha, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::flags(), and std::ios_base::goodbit.

4.891.4.2 template<typename _CharT , typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const [inline], [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2176 of file locale_facets.h.

4.891.4.3 template<typename _CharT , typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (iter_type *__beg*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, unsigned short & *__v*) const `[inline]`, `[protected]`, `[virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2181 of file locale_facets.h.

4.891.4.4 template<typename _CharT , typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (iter_type *__beg*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, unsigned int & *__v*) const `[inline]`, `[protected]`, `[virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2186 of file locale_facets.h.

4.891.4.5 template<typename _CharT , typename _InIter > virtual **iter_type std::num_get**< _CharT, _InIter >::do_get (**iter_type** *__beg*, **iter_type** *__end*, **ios_base** & *__io*, **ios_base::iostate** & *__err*, unsigned long & *__v*) const `[inline]`, `[protected]`, `[virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2191 of file locale_facets.h.

4.891.4.6 template<typename _CharT , typename _InIter > virtual **iter_type std::num_get**< _CharT, _InIter >::do_get (**iter_type** *__beg*, **iter_type** *__end*, **ios_base** & *__io*, **ios_base::iostate** & *__err*, long long & *__v*) const `[inline]`, `[protected]`, `[virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2197 of file locale_facets.h.

4.891.4.7 template< typename _CharT , typename _InIter > virtual iter_type std::num_get< _CharT, _InIter >::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const [inline], [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2202 of file locale_facets.h.

4.891.4.8 template< typename _CharT , typename _InIter > _InIter std::num_get< _CharT, _InIter >::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v) const [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 686 of file locale_facets.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

4.891.4.9 template<typename _CharT , typename _InIter > _InIter **std::num_get**< _CharT, _InIter >::do_get (**iter_type** *__beg,* **iter_type** *__end,* **ios_base** & *__io,* **ios_base::iostate** & *__err,* double & *__v*) const `[protected]`, `[virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 701 of file locale_facets.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

4.891.4.10 template<typename _CharT , typename _InIter > _InIter **std::num_get**< _CharT, _InIter >::do_get (**iter_type** *__beg,* **iter_type** *__end,* **ios_base** & *__io,* **ios_base::iostate** & *__err,* long double & *__v*) const `[protected]`, `[virtual]`

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 733 of file locale_facets.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::reserve().

4.891.4.11 template<typename _CharT , typename _InIter > _InIter std::num_get< _CharT, _InIter >::do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, void *& __v) const [protected], [virtual]

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See Also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 748 of file locale_facets.tcc.

References std::ios_base::basefield, std::ios_base::flags(), and std::ios_base::hex.

4.891.4.12 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, bool & __v) const [inline]

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling num_get::do_get().

If ios_base::boolalpha is set, attempts to read ctype<CharT>::truename() or ctype<CharT>::falsename(). Sets *v* to true or false if successful. Sets err to ios_base::failbit if reading the string fails. Sets err to ios_base::eofbit if the stream is emptied.

If ios_base::boolalpha is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set err to ios_base::failbit.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 1962 of file locale_facets.h.

Referenced by std::basic_istream< _CharT, _Traits >::operator>>().

4.891.4.13 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* long & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 1999 of file locale_facets.h.

4.891.4.14 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* unsigned short & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2004 of file locale_facets.h.

4.891.4.15 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* unsigned int & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2009 of file locale_facets.h.

4.891.4.16 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* unsigned long & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2014 of file locale_facets.h.

4.891.4.17 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* long long & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2020 of file locale_facets.h.

4.891.4.18 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* unsigned long long & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2025 of file locale_facets.h.

4.891.4.19 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, float & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2059 of file locale_facets.h.

4.891.4.20 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, double & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2064 of file locale_facets.h.

4.891.4.21 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* long double & *__v*) const [inline]

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2069 of file locale_facets.h.

4.891.4.22 template<typename _CharT , typename _InIter > iter_type std::num_get< _CharT, _InIter >::get (iter_type *__in,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* void *& *__v*) const [inline]

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling num_get::do_get().

The input characters are parsed like the scanf p specifier.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Definition at line 2102 of file locale_facets.h.

4.891.5 Member Data Documentation

4.891.5.1 template<typename _CharT , typename _InIter > **locale::id std::num_get< _CharT, _InIter >::id** `[static]`

Numpunct facet id.

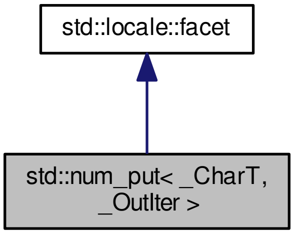
Definition at line 1926 of file locale_facets.h.

The documentation for this class was generated from the following files:

- locale_facets.h
- locale_facets.tcc

4.892 std::num_put< _CharT, _OutIter > Class Template Reference

Inheritance diagram for std::num_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type

Public Member Functions

- num_put (size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, bool __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const void ∗__v) const

- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const

- iter_type put (iter_type __s, ios_base &__io, char_type __fill, double __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long double __v) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼num_put ()
- void **_M_group_float** (const char ∗__grouping, size_t __grouping_size, char_type __sep, const char_type ∗__p, char_type ∗__new, char_type ∗__cs, int &__len) const
- void **_M_group_int** (const char ∗__grouping, size_t __grouping_size, char_type __sep, ios_base &__io, char_type ∗__new, char_type ∗__cs, int &__len) const
- template<typename _ValueT >
  iter_type **_M_insert_float** (iter_type, ios_base &__io, char_type __fill, char __mod, _ValueT __v) const
- template<typename _ValueT >
  iter_type **_M_insert_int** (iter_type, ios_base &__io, char_type __fill, _ValueT __v) const
- void **_M_pad** (char_type __fill, streamsize __w, ios_base &__io, char_type ∗__new, const char_type ∗__cs, int &__len) const

- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, bool __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, double) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, long double) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, const void ∗) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

4.892.1 Detailed Description

template<typename _CharT, typename _OutIter>class std::num_put< _CharT, _OutIter >

Primary class template num_put.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

The num_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_put facet.

Definition at line 2254 of file locale_facets.h.

4.892.2 Member Typedef Documentation

4.892.2.1 template<typename _CharT , typename _OutIter > typedef _CharT std::num_put< _CharT, _OutIter >::char_type

Public typedefs.

Definition at line 2260 of file locale_facets.h.

4.892.2.2 template<typename _CharT , typename _OutIter > typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type

Public typedefs.

Definition at line 2261 of file locale_facets.h.

4.892.3 Constructor & Destructor Documentation

4.892.3.1 template<typename _CharT , typename _OutIter > std::num_put< _CharT, _OutIter >::num_put (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 2275 of file locale_facets.h.

4.892.3.2 template<typename _CharT , typename _OutIter > virtual std::num_put< _CharT, _OutIter >::~num_put ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 2454 of file locale_facets.h.

4.892.4 Member Function Documentation

4.892.4.1 template<typename _CharT , typename _OutIter > _OutIter **std::num_put**< _CharT, _OutIter >::do_put (**iter_type** *__s,* **ios_base** & *__io,* **char_type** *__fill,* bool *__v*) const `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 1089 of file locale_facets.tcc.

References std::ios_base::_M_getloc(), std::ios_base::adjustfield, std::ios_base::boolalpha, std::ios_base::flags(), std::ios_base::left, and std::ios_base::width().

4.892.4.2 template<typename _CharT , typename _OutIter > virtual **iter_type std::num_put**< _CharT, _OutIter >::do_put (**iter_type** *__s,* **ios_base** & *__io,* **char_type** *__fill,* long *__v*) const `[inline]`, `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2474 of file locale_facets.h.

4.892.4.3 template<typename _CharT , typename _OutIter > virtual **iter_type std::num_put**< _CharT, _OutIter >::do_put (**iter_type** *__s,* **ios_base** & *__io,* **char_type** *__fill,* unsigned long *__v*) const `[inline]`, `[protected]`, `[virtual]`

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2478 of file locale_facets.h.

4.892.4.4 template<typename _CharT , typename _OutIter > virtual iter_type std::num_put< _CharT, _OutIter >::do_put (iter_type *__s,* ios_base & *__io,* char_type *__fill,* long long *__v*) const [inline], [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2484 of file locale_facets.h.

4.892.4.5 template<typename _CharT , typename _OutIter > virtual iter_type std::num_put< _CharT, _OutIter >::do_put (iter_type *__s,* ios_base & *__io,* char_type *__fill,* unsigned long long *__v*) const [inline], [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2489 of file locale_facets.h.

4.892.4.6 template<typename _CharT , typename _OutIter > _OutIter **std::num_put**< _CharT, _OutIter >::do_put (**iter_type** *__s,* **ios_base** & *__io,* **char_type** *__fill,* double *__v*) const [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 1141 of file locale_facets.tcc.

4.892.4.7 template<typename _CharT , typename _OutIter > _OutIter **std::num_put**< _CharT, _OutIter >::do_put (**iter_type** *__s,* **ios_base** & *__io,* **char_type** *__fill,* long double *__v*) const [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 1155 of file locale_facets.tcc.

4.892.4.8 template<typename _CharT , typename _OutIter > _OutIter **std::num_put**< _CharT, _OutIter >::do_put (**iter_type** *__s,* **ios_base** & *__io,* **char_type** *__fill,* const void * *__v*) const [protected], [virtual]

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 1162 of file locale_facets.tcc.

References std::ios_base::flags(), std::ios_base::hex, and std::ios_base::uppercase.

4.892.4.9 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type *__s,* ios_base & *__io,* char_type *__fill,* bool *__v*) const [inline]

Numeric formatting.

Formats the boolean *v* and inserts it into a stream. It does so by calling num_put::do_put().

If ios_base::boolalpha is set, writes ctype<CharT>::truename() or ctype<CharT>::falsename(). Otherwise formats *v* as an int.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2293 of file locale_facets.h.

4.892.4.10 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type *__s,* ios_base & *__io,* char_type *__fill,* long *__v*) const [inline]

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2335 of file locale_facets.h.

4.892.4.11 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base & __io, char_type __fill, unsigned long __v) const [inline]

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2339 of file locale_facets.h.

4.892.4.12 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type __s, ios_base & __io, char_type __fill, long long __v) const [inline]

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2345 of file locale_facets.h.

4.892.4.13 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type *__s*, ios_base & *__io*, char_type *__fill*, unsigned long long *__v*) const [inline]

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2349 of file locale_facets.h.

4.892.4.14 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type *__s*, ios_base & *__io*, char_type *__fill*, double *__v*) const [inline]

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::floatfield. If equal to ios_base::fixed, formats like the printf f specifier. Else if equal to ios_base::scientific, formats like e or E with ios_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.

The output precision is given by io.precision(). This precision is capped at numeric_limits::digits10 + 2 (different for double and long double). The default precision is 6.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showpoint is set, a decimal point will always be output.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2398 of file locale_facets.h.

4.892.4.15 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type *__s*, ios_base & *__io*, char_type *__fill*, long double *__v*) const [inline]

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::floatfield. If equal to ios_base::fixed, formats like the printf f specifier. Else if equal to ios_base::scientific, formats like e or E with ios_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.

The output precision is given by io.precision(). This precision is capped at numeric_limits::digits10 + 2 (different for double and long double). The default precision is 6.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showpoint is set, a decimal point will always be output.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2402 of file locale_facets.h.

4.892.4.16 template<typename _CharT , typename _OutIter > iter_type std::num_put< _CharT, _OutIter >::put (iter_type *__s,* ios_base & *__io,* char_type *__fill,* const void * *__v*) const [inline]

Numeric formatting.

Formats the pointer value *v* and inserts it into a stream. It does so by calling num_put::do_put().

This function formats *v* as an unsigned long with ios_base::hex and ios_base::showbase set.

Parameters

__s	Stream to write to.
__io	Source of locale and flags.
__fill	Char_type to use for filling.
__v	Value to format and insert.

Returns

Iterator after writing.

Definition at line 2423 of file locale_facets.h.

4.892.5 Member Data Documentation

4.892.5.1 template<typename _CharT , typename _OutIter > locale::id std::num_put< _CharT, _OutIter >::id [static]

Numpunct facet id.

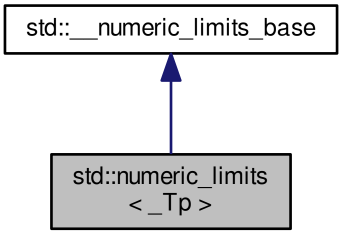
Definition at line 2265 of file locale_facets.h.

The documentation for this class was generated from the following files:

- locale_facets.h
- locale_facets.tcc

4.893 std::numeric_limits< _Tp > Struct Template Reference

Inheritance diagram for std::numeric_limits< _Tp >:



Static Public Member Functions

- static constexpr _Tp denorm_min () noexcept
- static constexpr _Tp epsilon () noexcept
- static constexpr _Tp infinity () noexcept
- static constexpr _Tp lowest () noexcept
- static constexpr _Tp max () noexcept
- static constexpr _Tp min () noexcept
- static constexpr _Tp quiet_NaN () noexcept
- static constexpr _Tp round_error () noexcept
- static constexpr _Tp signaling_NaN () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int digits10
- static constexpr float_denorm_style has_denorm
- static constexpr bool has_denorm_loss
- static constexpr bool has_infinity
- static constexpr bool has_quiet_NaN
- static constexpr bool has_signaling_NaN
- static constexpr bool is_bounded
- static constexpr bool is_exact
- static constexpr bool is_iec559
- static constexpr bool is_integer
- static constexpr bool is_modulo
- static constexpr bool is_signed
- static constexpr bool is_specialized
- static constexpr int max_digits10
- static constexpr int max_exponent
- static constexpr int max_exponent10

- static constexpr int min_exponent
- static constexpr int min_exponent10
- static constexpr int radix
- static constexpr float_round_style round_style
- static constexpr bool tinyness_before
- static constexpr bool traps

4.893.1 Detailed Description

template<typename _Tp>struct std::numeric_limits< _Tp >

Properties of fundamental types.

This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

_GLIBCXX_RESOLVE_LIB_DEFECTS: DRs 201 and 184 (hi Gaby!) are noted, but not incorporated in this documented (yet).

Definition at line 304 of file limits.

4.893.2 Member Function Documentation

4.893.2.1 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::denorm_min ( ) `[inline]`, `[static]`, `[noexcept]`

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

Definition at line 349 of file limits.

4.893.2.2 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::epsilon ( ) `[inline]`, `[static]`, `[noexcept]`

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

Definition at line 325 of file limits.

Referenced by std::binomial_distribution< _IntType >::operator()(), and std::poisson_distribution< _IntType >::operator()().

4.893.2.3 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::infinity ( ) `[inline]`, `[static]`, `[noexcept]`

The representation of positive infinity, if `has_infinity`.

Definition at line 333 of file limits.

4.893.2.4 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::lowest ( ) `[inline]`, `[static]`, `[noexcept]`

A finite value x such that there is no other finite value y where y < x.

Definition at line 319 of file limits.

4.893.2.5 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::max ( ) [inline], [static], [noexcept]

The maximum finite value.

Definition at line 313 of file limits.

Referenced by std::normal_distribution< result_type >::max(), std::lognormal_distribution< _RealType >::max(), std::gamma_distribution< result_type >::max(), std::chi_squared_distribution< _RealType >::max(), std::cauchy_distribution< _RealType >::max(), std::fisher_f_distribution< _RealType >::max(), std::student_t_distribution< _RealType >::max(), std::bernoulli_distribution::max(), std::geometric_distribution< _IntType >::max(), std::negative_binomial_distribution< _IntType >::max(), std::poisson_distribution< _IntType >::max(), std::exponential_distribution< _RealType >::max(), std::weibull_distribution< _RealType >::max(), std::extreme_value_distribution< _RealType >::max(), std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), std::binomial_distribution< _IntType >::operator()(), and std::poisson_distribution< _IntType >::operator()().

4.893.2.6 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::min ( ) [inline], [static], [noexcept]

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 309 of file limits.

Referenced by std::normal_distribution< result_type >::min(), std::cauchy_distribution< _RealType >::min(), std::student_t_distribution< _RealType >::min(), std::bernoulli_distribution::min(), std::extreme_value_distribution< _RealType >::min(), and std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()().

4.893.2.7 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN ( ) [inline], [static], [noexcept]

The representation of a quiet Not a Number, if `has_quiet_NaN`.

Definition at line 338 of file limits.

4.893.2.8 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::round_error ( ) [inline], [static], [noexcept]

The maximum rounding error measurement (see LIA-1).

Definition at line 329 of file limits.

4.893.2.9 template<typename _Tp > static constexpr _Tp std::numeric_limits< _Tp >::signaling_NaN ( ) [inline], [static], [noexcept]

The representation of a signaling Not a Number, if `has_signaling_NaN`.

Definition at line 343 of file limits.

4.893.3 Member Data Documentation

4.893.3.1 constexpr int std::__numeric_limits_base::digits [static], [inherited]

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 200 of file limits.

4.893.3.2 constexpr int std::__numeric_limits_base::digits10 `[static]`, `[inherited]`

The number of base 10 digits that can be represented without change.

Definition at line 203 of file limits.

4.893.3.3 constexpr float_denorm_style std::__numeric_limits_base::has_denorm `[static]`, `[inherited]`

See std::float_denorm_style for more information.

Definition at line 255 of file limits.

4.893.3.4 constexpr bool std::__numeric_limits_base::has_denorm_loss `[static]`, `[inherited]`

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 259 of file limits.

4.893.3.5 constexpr bool std::__numeric_limits_base::has_infinity `[static]`, `[inherited]`

True if the type has a representation for positive infinity.

Definition at line 244 of file limits.

4.893.3.6 constexpr bool std::__numeric_limits_base::has_quiet_NaN `[static]`, `[inherited]`

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 248 of file limits.

4.893.3.7 constexpr bool std::__numeric_limits_base::has_signaling_NaN `[static]`, `[inherited]`

True if the type has a representation for a signaling Not a Number.

Definition at line 252 of file limits.

4.893.3.8 constexpr bool std::__numeric_limits_base::is_bounded `[static]`, `[inherited]`

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 268 of file limits.

4.893.3.9 constexpr bool std::__numeric_limits_base::is_exact `[static]`, `[inherited]`

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 220 of file limits.

4.893.3.10 constexpr bool std::__numeric_limits_base::is_iec559 `[static]`, `[inherited]`

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 263 of file limits.

4.893.3.11 constexpr bool std::__numeric_limits_base::is_integer `[static]`, `[inherited]`

True if the type is integer.

Definition at line 215 of file limits.

4.893.3.12 constexpr bool std::__numeric_limits_base::is_modulo `[static]`, `[inherited]`

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or ∗ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 277 of file limits.

4.893.3.13 constexpr bool std::__numeric_limits_base::is_signed `[static]`, `[inherited]`

True if the type is signed.

Definition at line 212 of file limits.

4.893.3.14 constexpr bool std::__numeric_limits_base::is_specialized `[static]`, `[inherited]`

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 195 of file limits.

4.893.3.15 constexpr int std::__numeric_limits_base::max_digits10 `[static]`, `[inherited]`

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 208 of file limits.

4.893.3.16 constexpr int std::__numeric_limits_base::max_exponent `[static]`, `[inherited]`

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 237 of file limits.

4.893.3.17 constexpr int std::__numeric_limits_base::max_exponent10 `[static]`, `[inherited]`

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 241 of file limits.

4.893.3.18 constexpr int std::__numeric_limits_base::min_exponent `[static]`, `[inherited]`

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 228 of file limits.

4.893.3.19 constexpr int std::__numeric_limits_base::min_exponent10 `[static]`, `[inherited]`

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 232 of file limits.

4.893.3.20 constexpr int std::__numeric_limits_base::radix `[static]`, `[inherited]`

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 224 of file limits.

4.893.3.21 constexpr float_round_style std::__numeric_limits_base::round_style `[static]`, `[inherited]`

See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

Definition at line 288 of file limits.

4.893.3.22 constexpr bool std::__numeric_limits_base::tinyness_before `[static]`, `[inherited]`

True if tininess is detected before rounding. (see IEC 559)

Definition at line 283 of file limits.

4.893.3.23 constexpr bool std::__numeric_limits_base::traps `[static]`, `[inherited]`

True if trapping is implemented for this type.

Definition at line 280 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.894 std::numeric_limits< bool > Struct Template Reference

Static Public Member Functions

- static constexpr bool **denorm_min** () noexcept
- static constexpr bool **epsilon** () noexcept
- static constexpr bool **infinity** () noexcept
- static constexpr bool **lowest** () noexcept
- static constexpr bool **max** () noexcept
- static constexpr bool **min** () noexcept
- static constexpr bool **quiet_NaN** () noexcept
- static constexpr bool **round_error** () noexcept
- static constexpr bool **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**

- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.894.1 Detailed Description

template<>struct std::numeric_limits< bool >

numeric_limits<bool> specialization.

Definition at line 371 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.895 std::numeric_limits< char > Struct Template Reference

Static Public Member Functions

- static constexpr char **denorm_min** () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char **max** () noexcept
- static constexpr char **min** () noexcept
- static constexpr char **quiet_NaN** () noexcept
- static constexpr char **round_error** () noexcept
- static constexpr char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**

- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.895.1 Detailed Description

template<>struct std::numeric_limits< char >

numeric_limits<char> specialization.

Definition at line 440 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.896 std::numeric_limits< char16_t > Struct Template Reference

Static Public Member Functions

- static constexpr char16_t **denorm_min** () noexcept
- static constexpr char16_t **epsilon** () noexcept
- static constexpr char16_t **infinity** () noexcept
- static constexpr char16_t **lowest** () noexcept
- static constexpr char16_t **max** () noexcept
- static constexpr char16_t **min** () noexcept
- static constexpr char16_t **quiet_NaN** () noexcept
- static constexpr char16_t **round_error** () noexcept
- static constexpr char16_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**

- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.896.1 Detailed Description

template<>struct std::numeric_limits< char16_t >

numeric_limits<char16_t> specialization.

Definition at line 719 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.897 std::numeric_limits< char32_t > Struct Template Reference

Static Public Member Functions

- static constexpr char32_t **denorm_min** () noexcept
- static constexpr char32_t **epsilon** () noexcept
- static constexpr char32_t **infinity** () noexcept
- static constexpr char32_t **lowest** () noexcept
- static constexpr char32_t **max** () noexcept
- static constexpr char32_t **min** () noexcept
- static constexpr char32_t **quiet_NaN** () noexcept
- static constexpr char32_t **round_error** () noexcept
- static constexpr char32_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**

- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.897.1 Detailed Description

template<>struct std::numeric_limits< char32_t >

numeric_limits<char32_t> specialization.

Definition at line 780 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.898 std::numeric_limits< double > Struct Template Reference

Static Public Member Functions

- static constexpr double **denorm_min** () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double **max** () noexcept
- static constexpr double **min** () noexcept
- static constexpr double **quiet_NaN** () noexcept
- static constexpr double **round_error** () noexcept
- static constexpr double **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**

- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.898.1 Detailed Description

template<>struct std::numeric_limits< double >

numeric_limits<double> specialization.

Definition at line 1628 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.899 std::numeric_limits< float > Struct Template Reference

Static Public Member Functions

- static constexpr float **denorm_min** () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float **max** () noexcept
- static constexpr float **min** () noexcept
- static constexpr float **quiet_NaN** () noexcept
- static constexpr float **round_error** () noexcept
- static constexpr float **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**

- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.899.1 Detailed Description

template<>struct std::numeric_limits< float >

numeric_limits<float> specialization.

Definition at line 1553 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.900 std::numeric_limits< int > Struct Template Reference

Static Public Member Functions

- static constexpr int **denorm_min** () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int **max** () noexcept
- static constexpr int **min** () noexcept
- static constexpr int **quiet_NaN** () noexcept
- static constexpr int **round_error** () noexcept
- static constexpr int **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**

- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.900.1 Detailed Description

template<>struct std::numeric_limits< int >

numeric_limits<int> specialization.

Definition at line 982 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.901 std::numeric_limits< long > Struct Template Reference

Static Public Member Functions

- static constexpr long **denorm_min** () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long **max** () noexcept
- static constexpr long **min** () noexcept
- static constexpr long **quiet_NaN** () noexcept
- static constexpr long **round_error** () noexcept
- static constexpr long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**

- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.901.1 Detailed Description

template<>struct std::numeric_limits< long >

numeric_limits<long> specialization.

Definition at line 1121 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.902 std::numeric_limits< long double > Struct Template Reference

Static Public Member Functions

- static constexpr long double **denorm_min** () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double **min** () noexcept
- static constexpr long double **quiet_NaN** () noexcept
- static constexpr long double **round_error** () noexcept
- static constexpr long double **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**

- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.902.1 Detailed Description

template<>struct std::numeric_limits< long double >

numeric_limits<long double> specialization.

Definition at line 1703 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.903 std::numeric_limits< long long > Struct Template Reference

Static Public Member Functions

- static constexpr long long **denorm_min** () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long **quiet_NaN** () noexcept
- static constexpr long long **round_error** () noexcept
- static constexpr long long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**

- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.903.1 Detailed Description

template<>struct std::numeric_limits< long long >

numeric_limits<long long> specialization.

Definition at line 1261 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.904 std::numeric_limits< short > Struct Template Reference

Static Public Member Functions

- static constexpr short **denorm_min** () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short **max** () noexcept
- static constexpr short **min** () noexcept
- static constexpr short **quiet_NaN** () noexcept
- static constexpr short **round_error** () noexcept
- static constexpr short **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**

- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.904.1 Detailed Description

template<>struct std::numeric_limits< short >

numeric_limits<short> specialization.

Definition at line 842 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.905 std::numeric_limits< signed char > Struct Template Reference

Static Public Member Functions

- static constexpr signed char **denorm_min** () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char **quiet_NaN** () noexcept
- static constexpr signed char **round_error** () noexcept
- static constexpr signed char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.905.1 Detailed Description

template<>struct std::numeric_limits< signed char >

numeric_limits<signed char> specialization.

Definition at line 507 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.906 std::numeric_limits< unsigned char > Struct Template Reference

Static Public Member Functions

- static constexpr unsigned char **denorm_min** () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char **min** () noexcept
- static constexpr unsigned char **quiet_NaN** () noexcept
- static constexpr unsigned char **round_error** () noexcept
- static constexpr unsigned char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.906.1 Detailed Description

template<>struct std::numeric_limits< unsigned char >

numeric_limits<unsigned char> specialization.

Definition at line 577 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.907 std::numeric_limits< unsigned int > Struct Template Reference

Static Public Member Functions

- static constexpr unsigned int **denorm_min** () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int **quiet_NaN** () noexcept
- static constexpr unsigned int **round_error** () noexcept
- static constexpr unsigned int **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.907.1 Detailed Description

template<>struct std::numeric_limits< unsigned int >

numeric_limits<unsigned int> specialization.

Definition at line 1049 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.908 std::numeric_limits< unsigned long > Struct Template Reference

Static Public Member Functions

- static constexpr unsigned long **denorm_min** () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long **quiet_NaN** () noexcept
- static constexpr unsigned long **round_error** () noexcept
- static constexpr unsigned long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.908.1 Detailed Description

template<>struct std::numeric_limits< unsigned long >

numeric_limits<unsigned long> specialization.

Definition at line 1188 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.909 std::numeric_limits< unsigned long long > Struct Template Reference

Static Public Member Functions

- static constexpr unsigned long long **denorm_min** () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long **quiet_NaN** () noexcept
- static constexpr unsigned long long **round_error** () noexcept
- static constexpr unsigned long long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.909.1 Detailed Description

template<>struct std::numeric_limits< unsigned long long >

numeric_limits<unsigned long long> specialization.

Definition at line 1331 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.910 std::numeric_limits< unsigned short > Struct Template Reference

Static Public Member Functions

- static constexpr unsigned short **denorm_min** () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short **min** () noexcept
- static constexpr unsigned short **quiet_NaN** () noexcept
- static constexpr unsigned short **round_error** () noexcept
- static constexpr unsigned short **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.910.1 Detailed Description

template<>struct std::numeric_limits< unsigned short >

numeric_limits<unsigned short> specialization.

Definition at line 909 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.911 std::numeric_limits< wchar_t > Struct Template Reference

Static Public Member Functions

- static constexpr wchar_t **denorm_min** () noexcept
- static constexpr wchar_t **epsilon** () noexcept
- static constexpr wchar_t **infinity** () noexcept
- static constexpr wchar_t **lowest** () noexcept
- static constexpr wchar_t **max** () noexcept
- static constexpr wchar_t **min** () noexcept
- static constexpr wchar_t **quiet_NaN** () noexcept
- static constexpr wchar_t **round_error** () noexcept
- static constexpr wchar_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool **has_denorm_loss**
- static constexpr bool **has_infinity**
- static constexpr bool **has_quiet_NaN**
- static constexpr bool **has_signaling_NaN**
- static constexpr bool **is_bounded**
- static constexpr bool **is_exact**
- static constexpr bool **is_iec559**
- static constexpr bool **is_integer**
- static constexpr bool **is_modulo**
- static constexpr bool **is_signed**
- static constexpr bool **is_specialized**
- static constexpr int **max_digits10**
- static constexpr int **max_exponent**
- static constexpr int **max_exponent10**
- static constexpr int **min_exponent**
- static constexpr int **min_exponent10**
- static constexpr int **radix**
- static constexpr float_round_style **round_style**
- static constexpr bool **tinyness_before**
- static constexpr bool **traps**

4.911.1 Detailed Description

template<>struct std::numeric_limits< wchar_t >

numeric_limits<wchar_t> specialization.

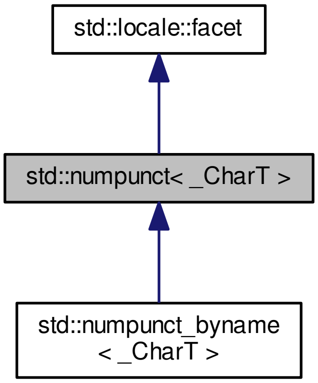
Definition at line 650 of file limits.

The documentation for this struct was generated from the following file:

- limits

4.912 std::numpunct< _CharT > Class Template Reference

Inheritance diagram for std::numpunct< _CharT >:



Public Types

- typedef __numpunct_cache< _CharT > **__cache_type**

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- numpunct (size_t __refs=0)
- numpunct (__cache_type *__cache, size_t __refs=0)
- numpunct (__c_locale __cloc, size_t __refs=0)
- char_type decimal_point () const
- string_type falsename () const
- string grouping () const
- char_type thousands_sep () const
- string_type truename () const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼numpunct ()
- void **_M_initialize_numpunct** (__c_locale __cloc=0)

- template<>
  void **_M_initialize_numpunct** (__c_locale __cloc)
- template<>
  void **_M_initialize_numpunct** (__c_locale __cloc)
- virtual char_type do_decimal_point () const
- virtual string_type do_falsename () const
- virtual string do_grouping () const
- virtual char_type do_thousands_sep () const
- virtual string_type do_truename () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __cache_type ∗ **_M_data**

4.912.1 Detailed Description

template<typename _CharT>class std::numpunct< _CharT >

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

Definition at line 1641 of file locale_facets.h.

4.912.2 Member Typedef Documentation

4.912.2.1 template<typename _CharT > typedef _CharT std::numpunct< _CharT >::char_type

Public typedefs.

Definition at line 1647 of file locale_facets.h.

4.912.2.2 template<typename _CharT > typedef basic_string<_CharT> std::numpunct< _CharT >::string_type

Public typedefs.

Definition at line 1648 of file locale_facets.h.

4.912.3 Constructor & Destructor Documentation

4.912.3.1 template<typename _CharT > std::numpunct< _CharT >::numpunct (size_t __refs = 0) [inline], [explicit]

Numpunct constructor.

Parameters

__refs	Refcount to pass to the base class.

Definition at line 1665 of file locale_facets.h.

4.912.3.2 template<typename _CharT > std::numpunct< _CharT >::numpunct (__cache_type * __cache, size_t __refs = 0) [inline], [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

Parameters

__cache	__numpunct_cache object.
__refs	Refcount to pass to the base class.

Definition at line 1679 of file locale_facets.h.

4.912.3.3 template<typename _CharT > std::numpunct< _CharT >::numpunct (__c_locale __cloc, size_t __refs = 0) [inline], [explicit]

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__refs	Refcount to pass to the base class.

Definition at line 1693 of file locale_facets.h.

4.912.3.4 template<typename _CharT > virtual std::numpunct< _CharT >::~numpunct ( ) [protected], [virtual]

Destructor.

4.912.4 Member Function Documentation

4.912.4.1 template<typename _CharT > char_type std::numpunct< _CharT >::decimal_point ( ) const [inline]

Return decimal point character.

This function returns a char_type to use as a decimal point. It does so by returning returning numpunct<char_type>::do_decimal_point().

Returns

char_type representing a decimal point.

Definition at line 1707 of file locale_facets.h.

4.912.4.2 template<typename _CharT > virtual char_type std::numpunct< _CharT >::do_decimal_point ( ) const `[inline]`, `[protected]`, `[virtual]`

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1794 of file locale_facets.h.

4.912.4.3 template<typename _CharT > virtual string_type std::numpunct< _CharT >::do_falsename ( ) const `[inline]`, `[protected]`, `[virtual]`

Return string representation of bool false.

Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Definition at line 1845 of file locale_facets.h.

4.912.4.4 template<typename _CharT > virtual string std::numpunct< _CharT >::do_grouping ( ) const `[inline]`, `[protected]`, `[virtual]`

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See Also

grouping() for details.

Returns

String representing grouping specification.

Definition at line 1819 of file locale_facets.h.

4.912.4.5 template<typename _CharT > virtual char_type std::numpunct< _CharT >::do_thousands_sep ( ) const `[inline]`, `[protected]`, `[virtual]`

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1806 of file locale_facets.h.

4.912.4.6 template<typename _CharT > virtual string_type std::numpunct< _CharT >::do_truename ( ) const [inline], [protected], [virtual]

Return string representation of bool true.

Returns a string_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Definition at line 1832 of file locale_facets.h.

4.912.4.7 template<typename _CharT > string_type std::numpunct< _CharT >::falsename ( ) const [inline]

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling numpunct<char_type>::do_falsename().

Returns

string_type representing printed form of false.

Definition at line 1777 of file locale_facets.h.

4.912.4.8 template<typename _CharT > string std::numpunct< _CharT >::grouping ( ) const [inline]

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling numpunct<char_type>::do_grouping().

Returns

string representing grouping specification.

Definition at line 1751 of file locale_facets.h.

4.912.4.9 template<typename _CharT > char_type std::numpunct< _CharT >::thousands_sep ( ) const [inline]

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning numpunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

Definition at line 1720 of file locale_facets.h.

4.912.4.10 template<typename _CharT > string_type std::numpunct< _CharT >::truename ( ) const [inline]

Return string representation of bool true.

This function returns a string_type containing the text representation for true bool variables. It does so by calling numpunct<char_type>::do_truename().

Returns

string_type representing printed form of true.

Definition at line 1764 of file locale_facets.h.

4.912.5 Member Data Documentation

4.912.5.1 template<typename _CharT > locale::id std::numpunct< _CharT >::id [static]

Numpunct facet id.

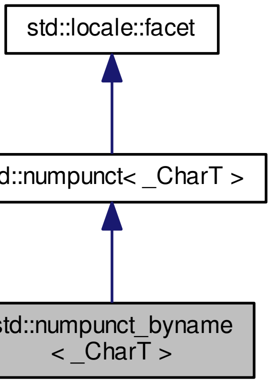
Definition at line 1657 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.913 std::numpunct_byname< _CharT > Class Template Reference

Inheritance diagram for std::numpunct_byname< _CharT >:



Public Types

- typedef __numpunct_cache< _CharT > **__cache_type**
- typedef _CharT **char_type**
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **numpunct_byname** (const char $*$__s, size_t __refs=0)
- char_type decimal_point () const
- string_type falsename () const
- string grouping () const
- char_type thousands_sep () const
- string_type truename () const

Static Public Attributes

- static locale::id id

Protected Member Functions

- void **_M_initialize_numpunct** (__c_locale __cloc=0)
- template<>
  void **_M_initialize_numpunct** (__c_locale __cloc)
- template<>
  void **_M_initialize_numpunct** (__c_locale __cloc)

- virtual char_type do_decimal_point () const
- virtual string_type do_falsename () const
- virtual string do_grouping () const
- virtual char_type do_thousands_sep () const
- virtual string_type do_truename () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __cache_type ∗ **_M_data**

4.913.1 Detailed Description

template<typename _CharT>class std::numpunct_byname< _CharT >

class numpunct_byname [22.2.3.2].

Definition at line 1874 of file locale_facets.h.

4.913.2 Member Function Documentation

4.913.2.1 template<typename _CharT > char_type std::numpunct< _CharT >::decimal_point ( ) const [inline], [inherited]

Return decimal point character.

This function returns a char_type to use as a decimal point. It does so by returning returning numpunct<char_type>::do_decimal_point().

Returns

char_type representing a decimal point.

Definition at line 1707 of file locale_facets.h.

4.913.2.2 template<typename _CharT > virtual char_type std::numpunct< _CharT >::do_decimal_point ( ) const [inline], [protected], [virtual], [inherited]

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Definition at line 1794 of file locale_facets.h.

4.913.2.3 template<typename _CharT > virtual string_type std::numpunct< _CharT >::do_falsename ( ) const [inline], [protected], [virtual], [inherited]

Return string representation of bool false.

Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Definition at line 1845 of file locale_facets.h.

4.913.2.4 template<typename _CharT > virtual string std::numpunct< _CharT >::do_grouping ( ) const [inline], [protected], [virtual], [inherited]

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See Also

grouping() for details.

Returns

String representing grouping specification.

Definition at line 1819 of file locale_facets.h.

4.913.2.5 template<typename _CharT > virtual char_type std::numpunct< _CharT >::do_thousands_sep ( ) const [inline], [protected], [virtual], [inherited]

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Definition at line 1806 of file locale_facets.h.

4.913.2.6 template<typename _CharT > virtual string_type std::numpunct< _CharT >::do_truename ( ) const [inline], [protected], [virtual], [inherited]

Return string representation of bool true.

Returns a string_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Definition at line 1832 of file locale_facets.h.

4.913.2.7 template<typename _CharT > string_type std::numpunct< _CharT >::falsename ( ) const [inline], [inherited]

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling numpunct<char_type>::do_falsename().

Returns

string_type representing printed form of false.

Definition at line 1777 of file locale_facets.h.

4.913.2.8 template<typename _CharT > string std::numpunct< _CharT >::grouping ( ) const [inline], [inherited]

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling numpunct<char_type>::do_grouping().

Returns

string representing grouping specification.

Definition at line 1751 of file locale_facets.h.

4.913.2.9 template<typename _CharT > char_type std::numpunct< _CharT >::thousands_sep ( ) const [inline], [inherited]

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning numpunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

Definition at line 1720 of file locale_facets.h.

4.913.2.10 template<typename _CharT > string_type std::numpunct< _CharT >::truename ( ) const [inline], [inherited]

Return string representation of bool true.

This function returns a string_type containing the text representation for true bool variables. It does so by calling numpunct<char_type>::do_truename().

Returns

string_type representing printed form of true.

Definition at line 1764 of file locale_facets.h.

4.913.3 Member Data Documentation

4.913.3.1 template<typename _CharT > **locale::id std::numpunct**< _CharT >::id [static], [inherited]

Numpunct facet id.

Definition at line 1657 of file locale_facets.h.

The documentation for this class was generated from the following file:

- locale_facets.h

4.914 std::once_flag Struct Reference

Public Member Functions

- constexpr once_flag () noexcept=default
- once_flag (const once_flag &)=delete
- once_flag & operator= (const once_flag &)=delete

Friends

- template<typename _Callable , typename... _Args>
  void call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)

4.914.1 Detailed Description

once_flag

Definition at line 723 of file mutex.

4.914.2 Constructor & Destructor Documentation

4.914.2.1 constexpr std::once_flag::once_flag ( ) [default], [noexcept]

Constructor.

4.914.2.2 std::once_flag::once_flag (const **once_flag** &) [delete]

Deleted copy constructor.

4.914.3 Member Function Documentation

4.914.3.1 **once_flag**& std::once_flag::operator= (const **once_flag** &) [delete]

Deleted assignment operator.

4.914.4 Friends And Related Function Documentation

4.914.4.1 template<typename _Callable , typename... _Args> void call_once (once_flag & *__once,* _Callable && *__f,* _Args &&... *__args*) [friend]

call_once

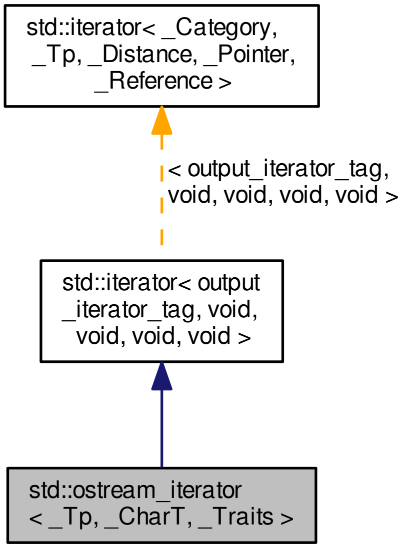
Definition at line 768 of file mutex.

The documentation for this struct was generated from the following file:

- mutex

4.915 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference

Inheritance diagram for std::ostream_iterator< _Tp, _CharT, _Traits >:



Public Types

- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

- typedef _CharT char_type
- typedef _Traits traits_type

- typedef basic_ostream< _CharT, _Traits > ostream_type

Public Member Functions

- ostream_iterator (ostream_type &__s)
- ostream_iterator (ostream_type &__s, const _CharT ∗__c)
- ostream_iterator (const ostream_iterator &__obj)
- ostream_iterator & **operator∗** ()
- ostream_iterator & **operator++** ()
- ostream_iterator & **operator++** (int)
- ostream_iterator & operator= (const _Tp &__value)

4.915.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>class std::ostream_iterator< _Tp, _CharT, _Traits >

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type Tp is the only type written by this iterator and there must be an operator<<(Tp) defined.

Template Parameters

_Tp	The type to write to the ostream.
_CharT	The ostream char_type.
_Traits	The ostream char_traits.

Definition at line 154 of file stream_iterator.h.

4.915.2 Member Typedef Documentation

4.915.2.1 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type

Public typedef.

Definition at line 160 of file stream_iterator.h.

4.915.2.2 typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.915.2.3 typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category [inherited]

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.915.2.4 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef **basic_ostream**<_CharT, _Traits> **std::ostream_iterator**< _Tp, _CharT, _Traits >**::ostream_type**

Public typedef.

Definition at line 162 of file stream_iterator.h.

4.915.2.5 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::pointer** `[inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.915.2.6 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::reference** `[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.915.2.7 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> typedef _Traits **std::ostream_iterator**< _Tp, _CharT, _Traits >**::traits_type**

Public typedef.

Definition at line 161 of file stream_iterator.h.

4.915.2.8 typedef void **std::iterator**< **output_iterator_tag** , void , void , void , void >**::value_type** `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.915.3 Constructor & Destructor Documentation

4.915.3.1 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> **std::ostream_iterator**< _Tp, _CharT, _Traits >**::ostream_iterator** (**ostream_type** & *__s*) `[inline]`

Construct from an ostream.

Definition at line 171 of file stream_iterator.h.

4.915.3.2 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> **std::ostream_iterator**< _Tp, _CharT, _Traits >**::ostream_iterator** (**ostream_type** & *__s*, const _CharT * *__c*) `[inline]`

Construct from an ostream.

The delimiter string *c* is written to the stream after every Tp written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

Parameters

__s	Underlying ostream to write to.
__c	CharT delimiter string to insert.

Definition at line 183 of file stream_iterator.h.

4.915.3.3 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> **std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator** (const ostream_iterator< _Tp, _CharT, _Traits > & *__obj*) [inline]

Copy constructor.

Definition at line 187 of file stream_iterator.h.

4.915.4 Member Function Documentation

4.915.4.1 template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>> **ostream_iterator& std::ostream_iterator< _Tp, _CharT, _Traits >::operator=** (const _Tp & *__value*) [inline]

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.

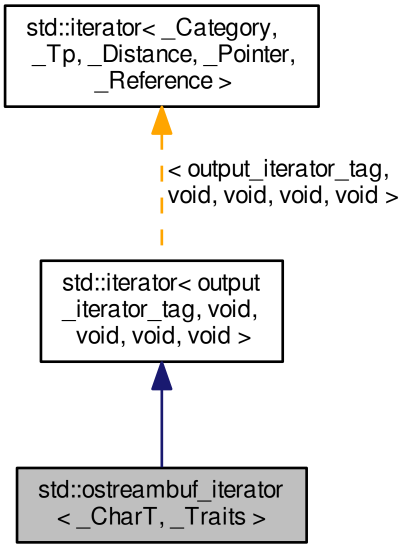
Definition at line 193 of file stream_iterator.h.

The documentation for this class was generated from the following file:

- stream_iterator.h

4.916 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference

Inheritance diagram for std::ostreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef void difference_type
- typedef output_iterator_tag iterator_category

- typedef void pointer
- typedef void reference
- typedef void value_type

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef basic_streambuf
  < _CharT, _Traits > streambuf_type
- typedef basic_ostream< _CharT,
  _Traits > ostream_type

Public Member Functions

- ostreambuf_iterator (ostream_type &__s) noexcept
- ostreambuf_iterator (streambuf_type ∗__s) noexcept
- ostreambuf_iterator & **_M_put** (const _CharT ∗__ws, streamsize __len)
- bool failed () const noexcept
- ostreambuf_iterator & operator∗ ()
- ostreambuf_iterator & operator++ (int)
- ostreambuf_iterator & operator++ ()
- ostreambuf_iterator & operator= (_CharT __c)

Friends

- template<typename _CharT2 >
  __gnu_cxx::__enable_if
  < __is_char< _CharT2 >
  ::__value, ostreambuf_iterator
  < _CharT2 > >::__type **copy** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)

4.916.1 Detailed Description

template<typename _CharT, typename _Traits>class std::ostreambuf_iterator< _CharT, _Traits >

Provides output iterator semantics for streambufs.

Definition at line 216 of file streambuf_iterator.h.

4.916.2 Member Typedef Documentation

4.916.2.1 template<typename _CharT , typename _Traits > typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type

Public typedefs.

Definition at line 223 of file streambuf_iterator.h.

4.916.2.2 typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type [inherited]

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.916.2.3 typedef **output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category** [inherited]

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.916.2.4 template<typename _CharT , typename _Traits > typedef **basic_ostream**<_CharT, _Traits> **std::ostreambuf_iterator**< _CharT, _Traits >**::ostream_type**

Public typedefs.

Definition at line 226 of file streambuf_iterator.h.

4.916.2.5 typedef void **std::iterator< output_iterator_tag** , void , void , void , void >**::pointer** [inherited]

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.916.2.6 typedef void **std::iterator< output_iterator_tag** , void , void , void , void >**::reference** [inherited]

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.916.2.7 template<typename _CharT , typename _Traits > typedef **basic_streambuf**<_CharT, _Traits> **std::ostreambuf_iterator**< _CharT, _Traits >**::streambuf_type**

Public typedefs.

Definition at line 225 of file streambuf_iterator.h.

4.916.2.8 template<typename _CharT , typename _Traits > typedef _Traits **std::ostreambuf_iterator**< _CharT, _Traits >**::traits_type**

Public typedefs.

Definition at line 224 of file streambuf_iterator.h.

4.916.2.9 typedef void **std::iterator< output_iterator_tag** , void , void , void , void >**::value_type** [inherited]

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.916.3 Constructor & Destructor Documentation

4.916.3.1 template<typename _CharT , typename _Traits > **std::ostreambuf_iterator**< _CharT, _Traits >**::ostreambuf_iterator** (**ostream_type** & *__s*) [inline], [noexcept]

Construct output iterator from ostream.

Definition at line 241 of file streambuf_iterator.h.

4.916.3.2 template<typename _CharT , typename _Traits > **std::ostreambuf_iterator**< _CharT, _Traits >**::ostreambuf_iterator** (**streambuf_type** ∗ *__s*) [inline], [noexcept]

Construct output iterator from streambuf.

Definition at line 245 of file streambuf_iterator.h.

4.916.4 Member Function Documentation

4.916.4.1 template<typename _CharT , typename _Traits > bool std::ostreambuf_iterator< _CharT, _Traits >::failed ( ) const [inline], [noexcept]

Return true if previous operator=() failed.

Definition at line 275 of file streambuf_iterator.h.

4.916.4.2 template<typename _CharT , typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator* ( ) [inline]

Return *this.

Definition at line 260 of file streambuf_iterator.h.

4.916.4.3 template<typename _CharT , typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++ (int) [inline]

Return *this.

Definition at line 265 of file streambuf_iterator.h.

4.916.4.4 template<typename _CharT , typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++ ( ) [inline]

Return *this.

Definition at line 270 of file streambuf_iterator.h.

4.916.4.5 template<typename _CharT , typename _Traits > ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator= (_CharT *__c*) [inline]

Write character to streambuf. Calls streambuf.sputc().

Definition at line 250 of file streambuf_iterator.h.

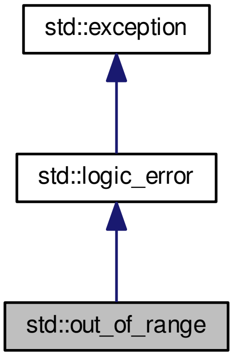
References std::basic_streambuf< _CharT, _Traits >::sputc().

The documentation for this class was generated from the following file:

- streambuf_iterator.h

4.917 std::out_of_range Class Reference

Inheritance diagram for std::out_of_range:



Public Member Functions

- **out_of_range** (const string &__arg)
- virtual const char ∗ what () const noexcept

4.917.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in basic_string).

Definition at line 100 of file stdexcept.

4.917.2 Member Function Documentation

4.917.2.1 virtual const char∗ std::logic_error::what ( ) const `[virtual]`, `[noexcept]`, `[inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

Reimplemented in std::future_error.

The documentation for this class was generated from the following file:

- stdexcept

4.918 std::output_iterator_tag Struct Reference

4.918.1 Detailed Description

Marking output iterators.

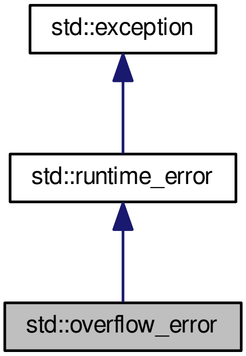
Definition at line 92 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.919 std::overflow_error Class Reference

Inheritance diagram for std::overflow_error:



Public Member Functions

- **overflow_error** (const string &__arg)
- virtual const char ∗ what () const noexcept

4.919.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 138 of file stdexcept.

4.919.2 Member Function Documentation

4.919.2.1 virtual const char∗ std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- stdexcept

4.920 std::owner_less< shared_ptr< _Tp > > Struct Template Reference

Inherits std::_Sp_owner_less< _Tp, _Tp1 >.

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const

4.920.1 Detailed Description

template<typename _Tp>struct std::owner_less< shared_ptr< _Tp > >

Partial specialization of owner_less for shared_ptr.

Definition at line 527 of file shared_ptr.h.

4.920.2 Member Typedef Documentation

4.920.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.920.2.2 typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.920.2.3 typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- shared_ptr.h

4.921 std::owner_less< weak_ptr< _Tp > > Struct Template Reference

Inherits std::_Sp_owner_less< _Tp, _Tp1 >.

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const

4.921.1 Detailed Description

template<typename _Tp>struct std::owner_less< weak_ptr< _Tp > >

Partial specialization of owner_less for weak_ptr.

Definition at line 533 of file shared_ptr.h.

4.921.2 Member Typedef Documentation

4.921.2.1 typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.921.2.2 typedef bool std::binary_function< _Tp , _Tp , bool >::result_type `[inherited]`

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.921.2.3 typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type `[inherited]`

`second_argument_type` is the type of the second argument

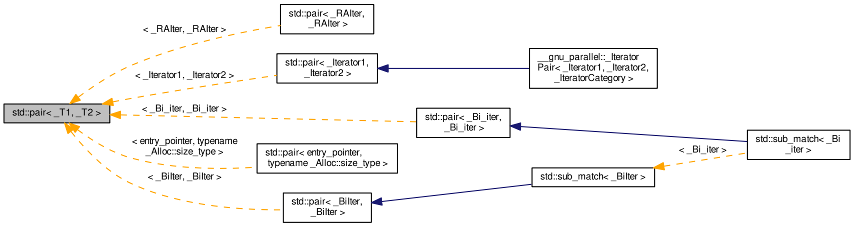
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- shared_ptr.h

4.922 std::pair< _T1, _T2 > Struct Template Reference

Inheritance diagram for std::pair< _T1, _T2 >:



Public Types

- typedef _T1 **first_type**
- typedef _T2 second_type

Public Member Functions

- constexpr pair ()
- constexpr pair (const _T1 &__a, const _T2 &__b)
- template<class _U1 , class _U2 , class = typename enable_if<__and_<is_convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type>
  constexpr pair (const pair< _U1, _U2 > &__p)
- constexpr **pair** (const pair &)=default
- constexpr **pair** (pair &&)=default
- template<class _U1 , class = typename enable_if<is_convertible<_U1, _T1>::value>::type>
  constexpr **pair** (_U1 &&__x, const _T2 &__y)
- template<class _U2 , class = typename enable_if<is_convertible<_U2, _T2>::value>::type>
  constexpr **pair** (const _T1 &__x, _U2 &&__y)
- template<class _U1 , class _U2 , class = typename enable_if<__and_<is_convertible<_U1, _T1>, is_convertible<_U2, _T2>>::value>::type>
  constexpr **pair** (_U1 &&__x, _U2 &&__y)
- template<class _U1 , class _U2 , class = typename enable_if<__and_<is_convertible<_U1, _T1>, is_convertible<_U2, _T2>>::value>::type>
  constexpr **pair** (pair< _U1, _U2 > &&__p)
- template<typename... _Args1, typename... _Args2>
  pair (piecewise_construct_t, tuple< _Args1...>, tuple< _Args2...>)
- void __p first && **noexcept** (swap(second, __p.second)))
- pair & **operator=** (const pair &__p)
- pair & **operator=** (pair &&__p) noexcept(__and_< is_nothrow_move_assignable< _T1 >
- template<class _U1 , class _U2 >
  pair & **operator=** (const pair< _U1, _U2 > &__p)
- template<class _U1 , class _U2 >
  pair & **operator=** (pair< _U1, _U2 > &&__p)
- void **swap** (pair &__p) noexcept(noexcept(swap(first

Public Attributes

- _T1 first
- pair
  is_nothrow_move_assignable
  < _T2 >::value **first**
- _T2 second
- **second**
- return ∗ **this**

4.922.1 Detailed Description

template<class _T1, class _T2>struct std::pair< _T1, _T2 >

Struct holding two objects of arbitrary type.

Template Parameters

_T1	Type of first object.
_T2	Type of second object.

Definition at line 96 of file stl_pair.h.

4.922.2 Member Typedef Documentation

4.922.2.1 template<class _T1, class _T2> typedef _T2 **std::pair< _T1, _T2 >::second_type**

`first_type` is the first bound type

Definition at line 99 of file stl_pair.h.

4.922.3 Constructor & Destructor Documentation

4.922.3.1 template<class _T1, class _T2> constexpr **std::pair< _T1, _T2 >::pair ( )** `[inline]`

`second` is a copy of the second object

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 108 of file stl_pair.h.

4.922.3.2 template<class _T1, class _T2> constexpr **std::pair< _T1, _T2 >::pair (const _T1 & *__a*, const _T2 & *__b*)** `[inline]`

Two objects may be passed to a `pair` constructor to be copied.

Definition at line 112 of file stl_pair.h.

4.922.3.3 template<class _T1, class _T2> template<class _U1 , class _U2 , class = typename enable_if<__and_<is_convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type> constexpr **std::pair< _T1, _T2 >::pair (const pair< _U1, _U2 > & *__p*)** `[inline]`

There is also a templated copy ctor for the `pair` class itself.

Definition at line 124 of file stl_pair.h.

4.922.4 Member Data Documentation

4.922.4.1 template<class _T1, class _T2> _T1 **std::pair**< _T1, _T2 >::first

`second_type` is the second bound type

Definition at line 101 of file stl_pair.h.

Referenced by std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer(), std::set< _StateIdT >::insert(), std::operator<(), and std::operator==().

4.922.4.2 template<class _T1, class _T2> _T2 **std::pair**< _T1, _T2 >::second

`first` is a copy of the first object

Definition at line 102 of file stl_pair.h.

Referenced by std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer(), std::set< _StateIdT >::insert(), and std::operator==().

The documentation for this struct was generated from the following files:

- stl_pair.h
- tuple

4.923 std::piecewise_constant_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- template<typename _InputIteratorB , typename _InputIteratorW >
  piecewise_constant_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
  piecewise_constant_distribution (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func >
  piecewise_constant_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **piecewise_constant_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- std::vector< double > densities () const
- std::vector< _RealType > intervals () const

- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_constant_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_constant_distribution< _RealType1 > &__x)

4.923.1 Detailed Description

template<typename _RealType = double>class std::piecewise_constant_distribution< _RealType >

A piecewise_constant_distribution random number distribution.

The formula for the piecewise constant probability mass function is

Definition at line 5480 of file random.h.

4.923.2 Member Typedef Documentation

4.923.2.1 template<typename _RealType = double> typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 5483 of file random.h.

4.923.3 Member Function Documentation

4.923.3.1 template<typename _RealType = double> std::vector<double> std::piecewise_constant_distribution< _RealType >::densities ( ) const [inline]

Returns a vector of the probability densities.

Definition at line 5601 of file random.h.

References std::vector< _Tp, _Alloc >::empty().

4.923.3.2 template<typename _RealType = double> std::vector<_RealType> std::piecewise_constant_distribution< _RealType >::intervals ( ) const [inline]

Returns a vector of the intervals.

Definition at line 5585 of file random.h.

References std::vector< _Tp, _Alloc >::empty().

4.923.3.3 template<typename _RealType = double> result_type std::piecewise_constant_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 5636 of file random.h.

References std::vector< _Tp, _Alloc >::back(), and std::vector< _Tp, _Alloc >::empty().

4.923.3.4 template<typename _RealType = double> result_type std::piecewise_constant_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 5626 of file random.h.

References std::vector< _Tp, _Alloc >::empty(), and std::vector< _Tp, _Alloc >::front().

4.923.3.5 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::piecewise_constant_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & __*urng*) [inline]

Generating functions.

Definition at line 5647 of file random.h.

References std::piecewise_constant_distribution< _RealType >::operator()().

Referenced by std::piecewise_constant_distribution< _RealType >::operator()().

4.923.3.6 template<typename _RealType = double> param_type std::piecewise_constant_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 5611 of file random.h.

4.923.3.7 template<typename _RealType = double> void std::piecewise_constant_distribution< _RealType >::param (const param_type & __*param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 5619 of file random.h.

4.923.3.8 template<typename _RealType = double> void std::piecewise_constant_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 5578 of file random.h.

4.923.4 Friends And Related Function Documentation

4.923.4.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > **std::basic_ostream**<_CharT, _Traits>& operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::piecewise_constant_distribution**< _RealType1 > & *__x*) `[friend]`

Inserts a piecewise_constan_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A piecewise_constan_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.923.4.2 template<typename _RealType = double> bool operator== (const **piecewise_constant_distribution**< _RealType > & *__d1*, const **piecewise_constant_distribution**< _RealType > & *__d2*) `[friend]`

Return true if two piecewise constant distributions have the same parameters.

Definition at line 5682 of file random.h.

4.923.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > **std::basic_istream**<_CharT, _Traits>& operator>> (**std::basic_istream**< _CharT, _Traits > & *__is*, **std::piecewise_constant_distribution**< _RealType1 > & *__x*) `[friend]`

Extracts a piecewise_constan_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A piecewise_constan_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.924 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef
  piecewise_constant_distribution
  < _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB , typename _InputIteratorW >
  param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
  param_type (initializer_list< _RealType > __bi, _Func __fw)
- template<typename _Func >
  param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **param_type** (const param_type &)=default
- std::vector< double > **densities** () const
- std::vector< _RealType > **intervals** () const
- param_type & **operator=** (const param_type &)=default

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **piecewise_constant_distribution< _RealType >**

4.924.1 Detailed Description

template<typename _RealType = double>struct std::piecewise_constant_distribution< _RealType >::param_type

Parameter type.

Definition at line 5489 of file random.h.

The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

4.925 std::piecewise_construct_t Struct Reference

4.925.1 Detailed Description

piecewise_construct_t

Definition at line 76 of file stl_pair.h.

The documentation for this struct was generated from the following file:

- stl_pair.h

4.926 std::piecewise_linear_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- template<typename _InputIteratorB , typename _InputIteratorW >
  piecewise_linear_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
  piecewise_linear_distribution (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func >
  piecewise_linear_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **piecewise_linear_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- std::vector< double > densities () const
- std::vector< _RealType > intervals () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_linear_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_linear_distribution &__d1, const piecewise_linear_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_linear_distribution< _RealType1 > &__x)

4.926.1 Detailed Description

template<typename _RealType = double>class std::piecewise_linear_distribution< _RealType >

A piecewise_linear_distribution random number distribution.

The formula for the piecewise linear probability mass function is

Definition at line 5747 of file random.h.

4.926.2 Member Typedef Documentation

4.926.2.1 template<typename _RealType = double> typedef _RealType std::piecewise_linear_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 5750 of file random.h.

4.926.3 Member Function Documentation

4.926.3.1 template<typename _RealType = double> std::vector<double> std::piecewise_linear_distribution< _RealType >::densities ( ) const [inline]

Return a vector of the probability densities of the distribution.

Definition at line 5871 of file random.h.

References std::vector< _Tp, _Alloc >::empty().

4.926.3.2 template<typename _RealType = double> std::vector<_RealType> std::piecewise_linear_distribution< _RealType >::intervals ( ) const [inline]

Return the intervals of the distribution.

Definition at line 5854 of file random.h.

References std::vector< _Tp, _Alloc >::empty().

4.926.3.3 template<typename _RealType = double> result_type std::piecewise_linear_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 5906 of file random.h.

References std::vector< _Tp, _Alloc >::back(), and std::vector< _Tp, _Alloc >::empty().

4.926.3.4 template<typename _RealType = double> result_type std::piecewise_linear_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 5896 of file random.h.

References std::vector< _Tp, _Alloc >::empty(), and std::vector< _Tp, _Alloc >::front().

4.926.3.5 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::piecewise_linear_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 5917 of file random.h.

References std::piecewise_linear_distribution< _RealType >::operator()().

Referenced by std::piecewise_linear_distribution< _RealType >::operator()().

4.926.3.6 template<typename _RealType = double> param_type std::piecewise_linear_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 5881 of file random.h.

4.926.3.7 template<typename _RealType = double> void std::piecewise_linear_distribution< _RealType >::param (const param_type & __param) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 5889 of file random.h.

4.926.3.8 template<typename _RealType = double> void std::piecewise_linear_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 5847 of file random.h.

4.926.4 Friends And Related Function Documentation

4.926.4.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::piecewise_linear_distribution< _RealType1 > & __x) [friend]

Inserts a piecewise_linear_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A piecewise_linear_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.926.4.2 template<typename _RealType = double> bool operator== (const piecewise_linear_distribution< _RealType > & __d1, const piecewise_linear_distribution< _RealType > & __d2) [friend]

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5952 of file random.h.

4.926.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & __is, std::piecewise_linear_distribution< _RealType1 > & __x) [friend]

Extracts a piecewise_linear_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A piecewise_linear_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.927 std::piecewise_linear_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef
  piecewise_linear_distribution
  < _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB , typename _InputIteratorW >
  param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- template<typename _Func >
  param_type (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func >
  param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- **param_type** (const param_type &)=default
- std::vector< double > **densities** () const
- std::vector< _RealType > **intervals** () const
- param_type & **operator=** (const param_type &)=default

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **piecewise_linear_distribution< _RealType >**

4.927.1 Detailed Description

template<typename _RealType = double>struct std::piecewise_linear_distribution< _RealType >::param_type

Parameter type.

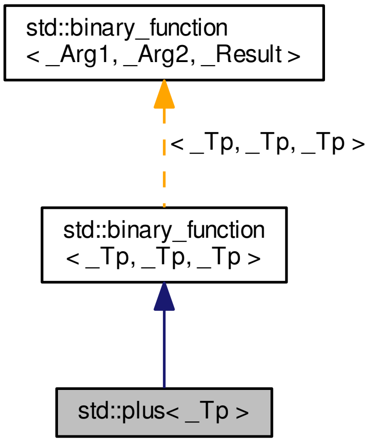
Definition at line 5756 of file random.h.

The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

4.928 std::plus< _Tp > Struct Template Reference

Inheritance diagram for std::plus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

4.928.1 Detailed Description

template<typename _Tp>struct std::plus< _Tp >

One of the math functors.

Definition at line 140 of file stl_function.h.

4.928.2 Member Typedef Documentation

4.928.2.1 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type `[inherited]`

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.928.2.2 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.928.2.3 typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

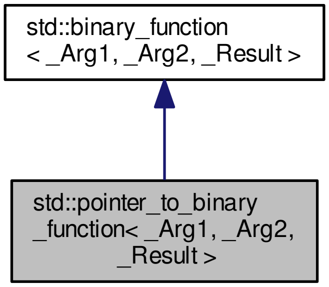
Definition at line 120 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.929 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference

Inheritance diagram for std::pointer_to_binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- **pointer_to_binary_function** (_Result(∗__x)(_Arg1, _Arg2))
- _Result **operator()** (_Arg1 __x, _Arg2 __y) const

Protected Attributes

- _Result(∗ **_M_ptr**)(_Arg1, _Arg2)

4.929.1 Detailed Description

template<typename _Arg1, typename _Arg2, typename _Result>class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >

One of the adaptors for function pointers.

Definition at line 447 of file stl_function.h.

4.929.2 Member Typedef Documentation

4.929.2.1 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type [inherited]

`first_argument_type` is the type of the first argument

Definition at line 117 of file stl_function.h.

4.929.2.2 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type [inherited]

`result_type` is the return type

Definition at line 123 of file stl_function.h.

4.929.2.3 template<typename _Arg1, typename _Arg2, typename _Result> typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]

`second_argument_type` is the type of the second argument

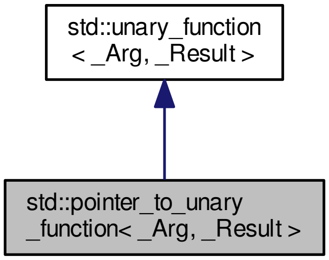
Definition at line 120 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.930 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference

Inheritance diagram for std::pointer_to_unary_function< _Arg, _Result >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

Public Member Functions

- **pointer_to_unary_function** (_Result(∗__x)(_Arg))
- _Result **operator()** (_Arg __x) const

Protected Attributes

- _Result(∗ **_M_ptr**)(_Arg)

4.930.1 Detailed Description

template<typename _Arg, typename _Result>class std::pointer_to_unary_function< _Arg, _Result >

One of the adaptors for function pointers.

Definition at line 422 of file stl_function.h.

4.930.2 Member Typedef Documentation

4.930.2.1 template<typename _Arg, typename _Result> typedef _Arg **std::unary_function< _Arg, _Result >::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.930.2.2 template<typename _Arg, typename _Result> typedef _Result **std::unary_function< _Arg, _Result >::result_type** `[inherited]`

`result_type` is the return type

Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.931 std::pointer_traits< _Ptr > Struct Template Reference

Inherits std::__ptrtr_pointer_to< _Ptr >.

Public Types

- typedef __ptrtr_diff_type
  < _Ptr >::__type difference_type
- typedef __ptrtr_elt_type< _Ptr >
  ::__type element_type

- typedef _Ptr pointer
- template<typename _Up >
  using **rebind** = typename __ptrtr_rebind< _Ptr, _Up >::__type

Static Public Member Functions

- static _Ptr **pointer_to** (__element_type &__e)

4.931.1 Detailed Description

template<typename _Ptr>struct std::pointer_traits< _Ptr >

Uniform interface to all pointer-like types.

Definition at line 137 of file ptr_traits.h.

4.931.2 Member Typedef Documentation

4.931.2.1 template<typename _Ptr > typedef __ptrtr_diff_type<_Ptr>::__type std::pointer_traits< _Ptr >::difference_type

Type used to represent the difference between two pointers.

Definition at line 144 of file ptr_traits.h.

4.931.2.2 template<typename _Ptr > typedef __ptrtr_elt_type<_Ptr>::__type std::pointer_traits< _Ptr >::element_type

The type pointed to.

Definition at line 142 of file ptr_traits.h.

4.931.2.3 template<typename _Ptr > typedef _Ptr std::pointer_traits< _Ptr >::pointer

The pointer type.

Definition at line 140 of file ptr_traits.h.

The documentation for this struct was generated from the following file:

- ptr_traits.h

4.932 std::pointer_traits< _Tp ∗ > Struct Template Reference

Public Types

- typedef ptrdiff_t difference_type
- typedef _Tp element_type
- typedef _Tp ∗ pointer
- template<typename _Up >
  using **rebind** = _Up ∗

Static Public Member Functions

- static pointer pointer_to (typename __ptrtr_not_void< element_type >::__type &__r) noexcept

4.932.1 Detailed Description

template<typename _Tp>struct std::pointer_traits< _Tp * >

Partial specialization for built-in pointers.

Definition at line 155 of file ptr_traits.h.

4.932.2 Member Typedef Documentation

4.932.2.1 template<typename _Tp > typedef ptrdiff_t **std::pointer_traits**< _Tp * >**::difference_type**

Type used to represent the difference between two pointers.

Definition at line 162 of file ptr_traits.h.

4.932.2.2 template<typename _Tp > typedef _Tp **std::pointer_traits**< _Tp * >**::element_type**

The type pointed to.

Definition at line 160 of file ptr_traits.h.

4.932.2.3 template<typename _Tp > typedef _Tp* **std::pointer_traits**< _Tp * >**::pointer**

The pointer type.

Definition at line 158 of file ptr_traits.h.

4.932.3 Member Function Documentation

4.932.3.1 template<typename _Tp > static **pointer std::pointer_traits**< _Tp * >::pointer_to (typename __ptrtr_not_void< **element_type** >::__type & *__r*) `[inline]`, `[static]`, `[noexcept]`

Obtain a pointer to an object.

Parameters

__r	A reference to an object of type `element_type`

Returns

`addressof(__r)`

Definition at line 173 of file ptr_traits.h.

References std::addressof().

The documentation for this struct was generated from the following file:

- ptr_traits.h

4.933 std::poisson_distribution< _IntType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **poisson_distribution** (double __mean=1.0)
- **poisson_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- double mean () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::poisson_distribution< _IntType1 > &__x)
- bool operator== (const poisson_distribution &__d1, const poisson_distribution &__d2)
- template<typename _IntType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::poisson_distribution< _IntType1 > &__x)

4.933.1 Detailed Description

template<typename _IntType = int>class std::poisson_distribution< _IntType >

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$ where μ is the parameter of the distribution.

Definition at line 4429 of file random.h.

4.933.2 Member Typedef Documentation

4.933.2.1 template<typename _IntType = int> typedef _IntType **std::poisson_distribution**< _IntType >::**result_type**

The type of the range of the distribution.

Definition at line 4432 of file random.h.

4.933.3 Member Function Documentation

4.933.3.1 template<typename _IntType = int> **result_type std::poisson_distribution**< _IntType >::max ( ) const `[inline]`

Returns the least upper bound value of the distribution.

Definition at line 4523 of file random.h.

References std::numeric_limits< _Tp >::max().

4.933.3.2 template<typename _IntType = int> double **std::poisson_distribution**< _IntType >::mean ( ) const `[inline]`

Returns the distribution parameter `mean`.

Definition at line 4494 of file random.h.

4.933.3.3 template<typename _IntType = int> **result_type std::poisson_distribution**< _IntType >::min ( ) const `[inline]`

Returns the greatest lower bound value of the distribution.

Definition at line 4516 of file random.h.

4.933.3.4 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > **result_type std::poisson_distribution**< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*) `[inline]`

Generating functions.

Definition at line 4531 of file random.h.

References std::poisson_distribution< _IntType >::operator()().

Referenced by std::poisson_distribution< _IntType >::operator()().

4.933.3.5 template<typename _IntType > template<typename _UniformRandomNumberGenerator > **poisson_distribution**< _IntType >::**result_type std::poisson_distribution**< _IntType >::operator() (_UniformRandomNumberGenerator & *__urng*, const **param_type** & *__param*)

A rejection algorithm when mean >= 12 and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if _GLIBCXX_USE_C99_MATH_TR1 is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1436 of file bits/random.tcc.

References std::abs(), std::numeric_limits< _Tp >::epsilon(), std::log(), and std::numeric_limits< _Tp >::max().

4.933.3.6 template<typename _IntType = int> **param_type std::poisson_distribution**< _IntType >::param ( ) const `[inline]`

Returns the parameter set of the distribution.

Definition at line 4501 of file random.h.

4.933.3.7 template<typename _IntType = int> void **std::poisson_distribution**< _IntType >::param (const **param_type** & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 4509 of file random.h.

4.933.3.8 template<typename _IntType = int> void **std::poisson_distribution**< _IntType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 4487 of file random.h.

References std::normal_distribution< _RealType >::reset().

4.933.4 Friends And Related Function Documentation

4.933.4.1 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > **std::basic_ostream**<_CharT, _Traits>& operator<< (**std::basic_ostream**< _CharT, _Traits > & *__os*, const **std::poisson_distribution**< _IntType1 > & *__x*) [friend]

Inserts a poisson_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A poisson_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.933.4.2 template<typename _IntType = int> bool operator== (const **poisson_distribution**< _IntType > & *__d1*, const **poisson_distribution**< _IntType > & *__d2*) [friend]

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4567 of file random.h.

4.933.4.3 template<typename _IntType = int> template<typename _IntType1 , typename _CharT , typename _Traits > **std::basic_istream**<_CharT, _Traits>& operator>> (**std::basic_istream**< _CharT, _Traits > & *__is*, **std::poisson_distribution**< _IntType1 > & *__x*) [friend]

Extracts a poisson_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A poisson_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.934 std::poisson_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef poisson_distribution
  < _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __mean=1.0)
- double **mean** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **poisson_distribution**< **_IntType** >

4.934.1 Detailed Description

template<typename _IntType = int>struct std::poisson_distribution< _IntType >::param_type

Parameter type.

Definition at line 4438 of file random.h.

The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

4.935 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- priority_queue (const _Compare &__x, const _Sequence &__s)
- **priority_queue** (const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence())
- template<typename _InputIterator >
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s)
- template<typename _InputIterator >
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare(), _Sequence &&__s=_Sequence())
- template<typename... _Args>
  void **emplace** (_Args &&...__args)
- bool empty () const
- void __pq c && **noexcept** (swap(comp, __pq.comp)))
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (priority_queue &__pq) noexcept(noexcept(swap(c
- const_reference top () const

Protected Attributes

- _Sequence **c**
- _Compare **comp**

4.935.1 Detailed Description

template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>class std::priority_queue< _Tp, _Sequence, _Compare >

A standard container automatically sorting its contents.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to vector<_Tp>.
_Compare	Comparison function object type, defaults to less<_Sequence::value_type>.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to std::vector, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as std::deque or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

Note

No equality/comparison operators are provided for priority_queue.
Sorting of the elements takes place as they are added to, and removed from, the priority_queue using the priority_queue's member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the priority_queue will not re-sort the elements for you. (How could it know to do so?)

Definition at line 367 of file stl_queue.h.

4.935.2 Constructor & Destructor Documentation

4.935.2.1 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (const _Compare & *__x*, const _Sequence & *__s*) [inline], [explicit]

Default constructor creates no elements.

Definition at line 402 of file stl_queue.h.

References std::make_heap().

4.935.2.2 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> template<typename _InputIterator > std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (_InputIterator *__first*, _InputIterator *__last*, const _Compare & *__x*, const _Sequence & *__s*) [inline]

Builds a queue from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__x	A comparison functor describing a strict weak ordering.
__s	An initial sequence with which to start.

Begins by copying *__s*, inserting a copy of the elements from [first,last) into the copy of *__s*, then ordering the copy according to *__x*.

For more information on function objects, see the documentation on functor base classes.

Definition at line 442 of file stl_queue.h.

References std::make_heap().

4.935.3 Member Function Documentation

4.935.3.1 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> bool std::priority_queue< _Tp, _Sequence, _Compare >::empty ( ) const [inline]

Returns true if the queue is empty.

Definition at line 468 of file stl_queue.h.

Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

4.935.3.2 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> void **std::priority_queue**< _Tp, _Sequence, _Compare >::pop ( ) `[inline]`

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 531 of file stl_queue.h.

References std::pop_heap().

Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

4.935.3.3 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> void **std::priority_queue**< _Tp, _Sequence, _Compare >::push (const value_type & *__x*) `[inline]`

Add data to the queue.

Parameters

__x	Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 496 of file stl_queue.h.

References std::push_heap().

Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

4.935.3.4 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> size_type **std::priority_queue**< _Tp, _Sequence, _Compare >::size ( ) const `[inline]`

Returns the number of elements in the queue.

Definition at line 473 of file stl_queue.h.

4.935.3.5 template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>> const_reference **std::priority_queue**< _Tp, _Sequence, _Compare >::top ( ) const `[inline]`

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 481 of file stl_queue.h.

Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

The documentation for this class was generated from the following file:

- stl_queue.h

4.936 std::queue< _Tp, _Sequence > Class Template Reference

Public Types

- typedef _Sequence::const_reference **const_reference**

- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- queue (const _Sequence &__c)
- **queue** (_Sequence &&__c=_Sequence())
- reference back ()
- const_reference back () const
- template<typename... _Args>
  void **emplace** (_Args &&...__args)
- bool empty () const
- reference front ()
- const_reference front () const
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (queue &__q) noexcept(noexcept(swap(c
- **swap** (c, __q.c)

Public Attributes

- void __q **c**

Protected Attributes

- _Sequence c

Friends

- template<typename _Tp1 , typename _Seq1 >
  bool **operator<** (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)
- template<typename _Tp1 , typename _Seq1 >
  bool **operator==** (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)

4.936.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>>class std::queue< _Tp, _Sequence >

A standard container giving FIFO behavior.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to deque<_Tp>.

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as std::list or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 93 of file stl_queue.h.

4.936.2 Constructor & Destructor Documentation

4.936.2.1 template<typename _Tp, typename _Sequence = deque<_Tp>> **std::queue**< _Tp, _Sequence >::**queue** (const _Sequence & *__c*) [inline], [explicit]

Default constructor creates no elements.

Definition at line 138 of file stl_queue.h.

4.936.3 Member Function Documentation

4.936.3.1 template<typename _Tp, typename _Sequence = deque<_Tp>> reference **std::queue**< _Tp, _Sequence >::back ( ) [inline]

Returns a read/write reference to the data at the last element of the queue.

Definition at line 185 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.2 template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference **std::queue**< _Tp, _Sequence >::back ( ) const [inline]

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 196 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.3 template<typename _Tp, typename _Sequence = deque<_Tp>> bool **std::queue**< _Tp, _Sequence >::empty ( ) const [inline]

Returns true if the queue is empty.

Definition at line 150 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.4 template<typename _Tp, typename _Sequence = deque<_Tp>> reference **std::queue**< _Tp, _Sequence >::front ( ) [inline]

Returns a read/write reference to the data at the first element of the queue.

Definition at line 163 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.5 template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::queue< _Tp, _Sequence >::front ( ) const [inline]

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 174 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.6 template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue< _Tp, _Sequence >::pop ( ) [inline]

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 238 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.7 template<typename _Tp, typename _Sequence = deque<_Tp>> void std::queue< _Tp, _Sequence >::push (const value_type & __x) [inline]

Add data to the end of the queue.

Parameters

__x	Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 212 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.3.8 template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::queue< _Tp, _Sequence >::size ( ) const [inline]

Returns the number of elements in the queue.

Definition at line 155 of file stl_queue.h.

References std::queue< _Tp, _Sequence >::c.

4.936.4 Member Data Documentation

4.936.4.1 template<typename _Tp, typename _Sequence = deque<_Tp>> _Sequence std::queue< _Tp, _Sequence >::c [protected]

'c' is the underlying container. Maintainers wondering why this isn't uglified as per style guidelines should note that this name is specified in the standard, [23.2.3.1]. (Why? Presumably for the same reason that it's protected instead of private: to allow derivation. But none of the other containers allow for derivation. Odd.)

Definition at line 126 of file stl_queue.h.

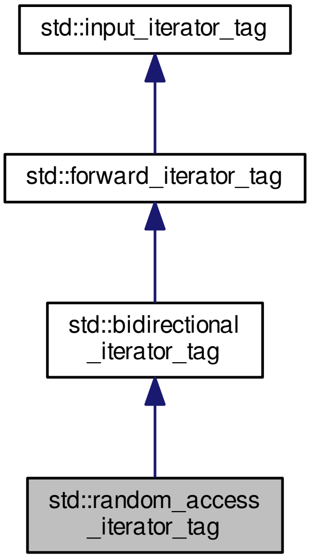
Referenced by std::queue< _Tp, _Sequence >::back(), std::queue< _Tp, _Sequence >::empty(), std::queue< _Tp, _Sequence >::front(), std::operator<(), std::operator==(), std::queue< _Tp, _Sequence >::pop(), std::queue< _Tp, _Sequence >::push(), and std::queue< _Tp, _Sequence >::size().

The documentation for this class was generated from the following file:

- stl_queue.h

4.937 std::random_access_iterator_tag Struct Reference

Inheritance diagram for std::random_access_iterator_tag:



4.937.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations.

Definition at line 103 of file stl_iterator_base_types.h.

The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

4.938 std::random_device Class Reference

Public Types

- typedef unsigned int result_type

Public Member Functions

- **random_device** (const std::string &__token="mt19937")
- **random_device** (const random_device &)=delete
- double **entropy** () const noexcept
- result_type **operator()** ()
- void **operator=** (const random_device &)=delete

Static Public Member Functions

- static constexpr result_type **max** ()
- static constexpr result_type **min** ()

4.938.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1575 of file random.h.

4.938.2 Member Typedef Documentation

4.938.2.1 typedef unsigned int **std::random_device::result_type**

The type of the generated random value.

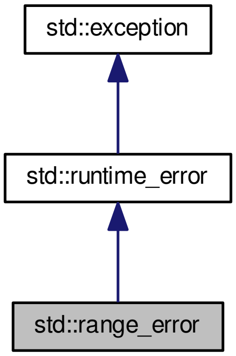
Definition at line 1579 of file random.h.

The documentation for this class was generated from the following file:

- random.h

4.939 std::range_error Class Reference

Inheritance diagram for std::range_error:



Public Member Functions

- **range_error** (const string &__arg)
- virtual const char ∗ what () const noexcept

4.939.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 130 of file stdexcept.

4.939.2 Member Function Documentation

4.939.2.1 virtual const char∗ std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

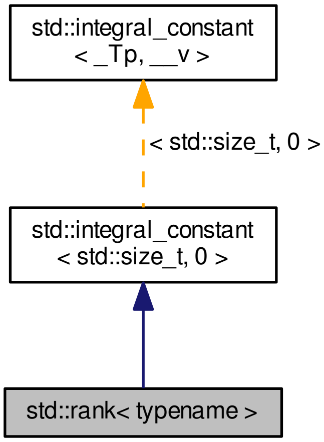
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- stdexcept

4.940 std::rank< typename > Struct Template Reference

Inheritance diagram for std::rank< typename >:



Public Types

- typedef integral_constant
  < std::size_t, __v > **type**

- typedef std::size_t **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr std::size_t **value**

4.940.1 Detailed Description

template<typename>struct std::rank< typename >

rank

Definition at line 1243 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.941 std::ratio< _Num, _Den > Struct Template Reference

Public Types

- typedef ratio< num, den > **type**

Static Public Attributes

- static constexpr intmax_t **den**
- static constexpr intmax_t **num**

4.941.1 Detailed Description

template<intmax_t _Num, intmax_t _Den = 1>struct std::ratio< _Num, _Den >

Provides compile-time rational arithmetic.

This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type intmax_t. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

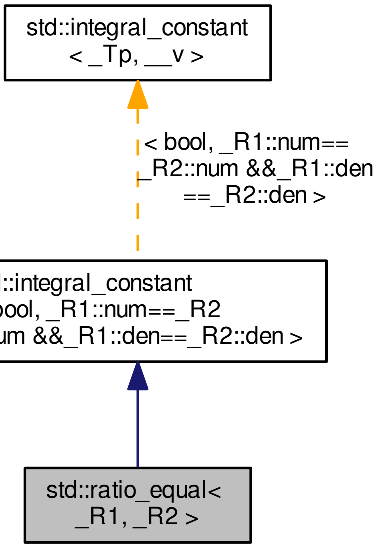
Definition at line 263 of file ratio.

The documentation for this struct was generated from the following file:

- ratio

4.942 std::ratio_equal< _R1, _R2 > Struct Template Reference

Inheritance diagram for std::ratio_equal< _R1, _R2 >:



Public Types

- typedef integral_constant < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.942.1 Detailed Description

template<typename _R1, typename _R2>struct std::ratio_equal< _R1, _R2 >

ratio_equal

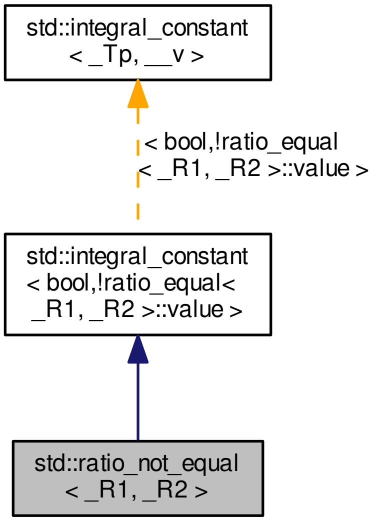
Definition at line 340 of file ratio.

The documentation for this struct was generated from the following file:

- ratio

4.943 std::ratio_not_equal< _R1, _R2 > Struct Template Reference

Inheritance diagram for std::ratio_not_equal< _R1, _R2 >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.943.1 Detailed Description

template<typename _R1, typename _R2>struct std::ratio_not_equal< _R1, _R2 >

ratio_not_equal

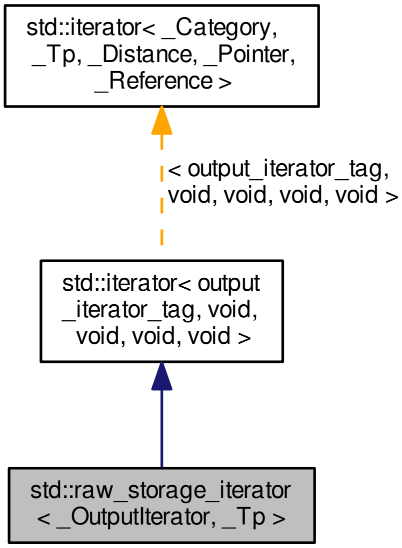
Definition at line 346 of file ratio.

The documentation for this struct was generated from the following file:

- ratio

4.944 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference

Inheritance diagram for std::raw_storage_iterator< _OutputIterator, _Tp >:



Public Types

- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- **raw_storage_iterator** (_OutputIterator __x)
- raw_storage_iterator & **operator*** ()
- raw_storage_iterator < _OutputIterator, _Tp > & **operator++** ()
- raw_storage_iterator < _OutputIterator, _Tp > **operator++** (int)
- raw_storage_iterator & **operator=** (const _Tp &__element)

Protected Attributes

- _OutputIterator **_M_iter**

4.944.1 Detailed Description

template< class _OutputIterator, class _Tp > class std::raw_storage_iterator< _OutputIterator, _Tp >

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 68 of file stl_raw_storage_iter.h.

4.944.2 Member Typedef Documentation

4.944.2.1 typedef void **std::iterator< output_iterator_tag , void , void , void , void >::difference_type** `[inherited]`

Distance between iterators is represented as this type.

Definition at line 125 of file stl_iterator_base_types.h.

4.944.2.2 typedef **output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >::iterator_category** `[inherited]`

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.944.2.3 typedef void **std::iterator< output_iterator_tag , void , void , void , void >::pointer** `[inherited]`

This type represents a pointer-to-value_type.

Definition at line 127 of file stl_iterator_base_types.h.

4.944.2.4 typedef void **std::iterator< output_iterator_tag , void , void , void , void >::reference** `[inherited]`

This type represents a reference-to-value_type.

Definition at line 129 of file stl_iterator_base_types.h.

4.944.2.5 typedef void **std::iterator< output_iterator_tag , void , void , void , void >::value_type** `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

The documentation for this class was generated from the following file:

- stl_raw_storage_iter.h

4.945 std::recursive_mutex Class Reference

Inherits std::__recursive_mutex_base.

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **recursive_mutex** (const recursive_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** ()
- recursive_mutex & **operator=** (const recursive_mutex &)=delete
- bool **try_lock** () noexcept
- void **unlock** ()

Private Types

- typedef __gthread_recursive_mutex_t **__native_type**

Private Attributes

- __native_type **_M_mutex**

4.945.1 Detailed Description

recursive_mutex

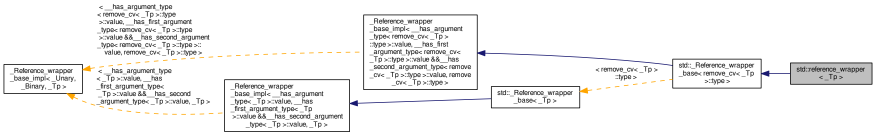
Definition at line 160 of file mutex.

The documentation for this class was generated from the following file:

- mutex

4.946 std::reference_wrapper< _Tp > Class Template Reference

Inheritance diagram for std::reference_wrapper< _Tp >:



Public Types

- typedef _Tp **type**

Public Member Functions

- **reference_wrapper** (_Tp &__indata) noexcept
- **reference_wrapper** (_Tp &&)=delete
- **reference_wrapper** (const reference_wrapper< _Tp > &__inref) noexcept
- _Tp & **get** () const noexcept
- **operator _Tp &** () const noexcept

- template<typename... _Args>
  result_of< _Tp &(_Args &&...)>
  ::type **operator()** (_Args &&... __args) const
- reference_wrapper & **operator=** (const reference_wrapper< _Tp > &__inref) noexcept

4.946.1 Detailed Description

template<typename _Tp>class std::reference_wrapper< _Tp >

Primary class template for reference_wrapper.

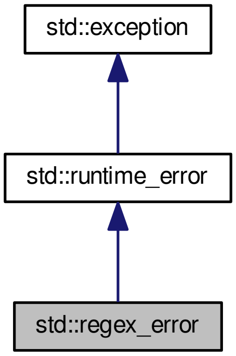
Definition at line 431 of file functional.

The documentation for this class was generated from the following file:

- functional

4.947 std::regex_error Class Reference

Inheritance diagram for std::regex_error:



Public Member Functions

- regex_error (regex_constants::error_type __ecode)
- regex_constants::error_type code () const
- virtual const char ∗ what () const noexcept

4.947.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 136 of file regex_error.h.

4.947.2 Constructor & Destructor Documentation

4.947.2.1 std::regex_error::regex_error (**regex_constants::error_type** *__ecode*) `[explicit]`

Constructs a regex_error object.

Parameters

__ecode	the regex error code.

4.947.3 Member Function Documentation

4.947.3.1 **regex_constants::error_type** std::regex_error::code ( ) const `[inline]`

Gets the regex error code.

Returns

the regex error code.

Definition at line 157 of file regex_error.h.

4.947.3.2 virtual const char∗ std::runtime_error::what ( ) const `[virtual], [noexcept], [inherited]`

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- regex_error.h

4.948 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const value_type ∗ **pointer**
- typedef const value_type & **reference**
- typedef basic_regex< _Ch_type, _Rx_traits > **regex_type**
- typedef match_results< _Bi_iter > **value_type**

Public Member Functions

- regex_iterator ()
- regex_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_iterator (const regex_iterator &__rhs)
- bool operator!= (const regex_iterator &__rhs)
- const value_type & operator∗ ()
- regex_iterator & operator++ ()

- regex_iterator operator++ (int)
- const value_type ∗ operator-> ()
- regex_iterator & operator= (const regex_iterator &__rhs)
- bool operator== (const regex_iterator &__rhs)

4.948.1 Detailed Description

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >

An iterator adaptor that will provide repeated calls of regex_search over a range until no more matches remain.

Definition at line 2204 of file regex.h.

4.948.2 Constructor & Destructor Documentation

4.948.2.1 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ( )

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Todo Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

4.948.2.2 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (_Bi_iter *__a*, _Bi_iter *__b*, const regex_type & *__re*, regex_constants::match_flag_type *__m = regex_constants::match_default*)

Constructs a regex_iterator...

Parameters

__a	[IN] The start of a text range to search.
__b	[IN] One-past-the-end of the text range to search.
__re	[IN] The regular expression to match.
__m	[IN] Policy flags for match rules.

Todo Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

4.948.2.3 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Copy constructs a regex_iterator.

Todo Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

4.948.3 Member Function Documentation

4.948.3.1 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!= (const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.948.3.2 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ( )

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.948.3.3 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ( )

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.948.3.4 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (int)

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.948.3.5 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const value_type* std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ( )

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.948.3.6 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.948.3.7 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

The documentation for this class was generated from the following file:

- regex.h

4.949 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const value_type ∗ **pointer**
- typedef const value_type & **reference**
- typedef basic_regex< _Ch_type, _Rx_traits > **regex_type**
- typedef sub_match< _Bi_iter > **value_type**

Public Member Functions

- regex_token_iterator ()
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector< int > &__submatches, regex_constants::match_flag_type __m=regex_constants::match_default)
- template<std::size_t _Nm>
  regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[_Nm], regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (const regex_token_iterator &__rhs)
- bool operator!= (const regex_token_iterator &__rhs)
- const value_type & operator∗ ()
- regex_token_iterator & operator++ ()
- regex_token_iterator operator++ (int)
- const value_type ∗ operator-> ()
- regex_token_iterator & operator= (const regex_token_iterator &__rhs)
- bool operator== (const regex_token_iterator &__rhs)

4.949.1 Detailed Description

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a std::sub_match object.

Definition at line 2318 of file regex.h.

4.949.2 Constructor & Destructor Documentation

4.949.2.1 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ( )**

Default constructs a regex_token_iterator.

Todo Implement this function.

A default-constructed regex_token_iterator is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

4.949.2.2 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (** _Bi_iter *__a,* _Bi_iter *__b,* const **regex_type** & *__re,* int *__submatch =* 0*,* **regex_constants::match_flag_type** *__m =* **regex_constants::match_default)**

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatch	[IN] Which submatch to return. There are some special values for this parameter: • -1 each enumerated subexpression does NOT match the regular expression (aka field splitting) • 0 the entire string matching the subexpression is returned for each match within the text. • >0 enumerates only the indicated subexpression from a match within the text.
__m	[IN] Policy flags for match rules.

Todo Implement this function.

Needs documentation! See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html`

4.949.2.3 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (** _Bi_iter *__a,* _Bi_iter *__b,* const **regex_type** & *__re,* const **std::vector**< int > & *__submatches,* **regex_constants::match_flag_type** *__m =* **regex_constants::match_default)**

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.949.2.4 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> template<std::size_t _Nm> **std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::**regex_token_iterator** (_Bi_iter *__a,* _Bi_iter *__b,* const **regex_type** & *__re,* const int(&) *__submatches[_Nm],* **regex_constants::match_flag_type** *__m* = **regex_constants::match_default**)

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

Todo Implement this function.

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-_style.html

4.949.2.5 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::**regex_token_iterator** (const **regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Copy constructs a regex_token_iterator.

Parameters

__rhs	[IN] A regex_token_iterator to copy.

Todo Implement this function.

4.949.3 Member Function Documentation

4.949.3.1 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool **std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator!= (const **regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Compares a regex_token_iterator to another for inequality.

Todo Implement this function.

4.949.3.2 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const **value_type**& **std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator∗ ( )

Dereferences a regex_token_iterator.

Todo Implement this function.

4.949.3.3 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **regex_token_iterator& std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ( )

Increments a regex_token_iterator.

Todo Implement this function.

4.949.3.4 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **regex_token_iterator std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (int)

Postincrements a regex_token_iterator.

Todo Implement this function.

4.949.3.5 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> const **value_type**∗ **std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ( )

Selects a regex_token_iterator member.

Todo Implement this function.

4.949.3.6 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> **regex_token_iterator& std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator= (const **regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Assigns a regex_token_iterator to another.

Parameters

__rhs	[IN] A regex_token_iterator to copy.

Todo Implement this function.

4.949.3.7 template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>> bool **std::regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits >::operator== (const **regex_token_iterator**< _Bi_iter, _Ch_type, _Rx_traits > & *__rhs*)

Compares a regex_token_iterator to another for equality.

Todo Implement this function.

The documentation for this class was generated from the following file:

- regex.h

4.950 std::regex_traits< _Ch_type > Struct Template Reference

Public Types

- typedef std::ctype_base::mask **char_class_type**
- typedef _Ch_type **char_type**
- typedef std::locale **locale_type**
- typedef std::basic_string
  < char_type > **string_type**

Public Member Functions

- regex_traits ()
- locale_type getloc () const
- locale_type imbue (locale_type __loc)
- bool isctype (_Ch_type __c, char_class_type __f) const
- template<typename _Fwd_iter >
  char_class_type lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const
- template<typename _Fwd_iter >
  string_type lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter >
  string_type transform (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter >
  string_type transform_primary (_Fwd_iter __first, _Fwd_iter __last) const
- char_type translate (char_type __c) const
- char_type translate_nocase (char_type __c) const
- int value (_Ch_type __ch, int __radix) const

Static Public Member Functions

- static std::size_t length (const char_type ∗__p)

Protected Attributes

- locale_type **_M_locale**

4.950.1 Detailed Description

template<typename _Ch_type>struct std::regex_traits< _Ch_type >

Class regex_traits. Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 51 of file regex.h.

4.950.2 Constructor & Destructor Documentation

4.950.2.1 template<typename _Ch_type > std::regex_traits< _Ch_type >::regex_traits ( ) [inline]

Constructs a default traits object.

Definition at line 63 of file regex.h.

4.950.3 Member Function Documentation

4.950.3.1 template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::getloc ( ) const [inline]

Gets a copy of the current locale in use by the regex_traits object.

Definition at line 270 of file regex.h.

4.950.3.2 template<typename _Ch_type > locale_type std::regex_traits< _Ch_type >::imbue (locale_type __loc) [inline]

Imbues the regex_traits object with a copy of a new locale.

Parameters

__loc	A locale.

Returns

a copy of the previous locale in use by the regex_traits object.

Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by *this.

Definition at line 259 of file regex.h.

4.950.3.3 template<typename _Ch_type > static std::size_t std::regex_traits< _Ch_type >::length (const char_type * __p) [inline], [static]

Gives the length of a C-style string starting at `__p`.

Parameters

__p	a pointer to the start of a character sequence.

Returns

the number of characters between `*__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 76 of file regex.h.

4.950.3.4 template<typename _Ch_type > template<typename _Fwd_iter > char_class_type std::regex_traits< _Ch_type >::lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase = false) const [inline]

Maps one or more characters to a named character classification.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.
__icase	ignores the case of the classification name.

Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range [__first, __last). If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, [[:lower:]] is the same as [[:alpha:]]), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- d
- w
- s
- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower
- print
- punct
- space
- upper
- xdigit

Todo Implement this function.

Definition at line 215 of file regex.h.

4.950.3.5 template< typename _Ch_type > template< typename _Fwd_iter > string_type std::regex_traits< _Ch_type >::lookup_collatename (_Fwd_iter *__first,* _Fwd_iter *__last*) const [inline]

Gets a collation element by name.

Parameters

__first	beginning of the collation element name.
__last	one-past-the-end of the collation element name.

Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [__first, __last). Returns an empty string if the character sequence is not a valid collating element.

Todo Implement this function.

Definition at line 171 of file regex.h.

4.950.3.6 template<typename _Ch_type > template<typename _Fwd_iter > **string_type std::regex_traits< _Ch_type >::transform** (_Fwd_iter *__first,* _Fwd_iter *__last*) const [inline]

Gets a sort key for a character sequence.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then v.transform(G1, G2) < v.transform(H1, H2).

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

Returns

a locale-specific sort key equivalent to the input range.

Exceptions

std::bad_cast	if the current locale does not have a collate facet.

Definition at line 129 of file regex.h.

References std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

4.950.3.7 template<typename _Ch_type > template<typename _Fwd_iter > **string_type std::regex_traits< _Ch_type >::transform_primary** (_Fwd_iter *__first,* _Fwd_iter *__last*) const [inline]

Gets a sort key for a character sequence, independent of case.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.

Effects: if typeid(use_facet<collate<_Ch_type> >) == typeid(collate_byname<_Ch_type>) and the form of the sort key returned by collate_byname<_Ch_type>::transform(__first, __last) is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

Todo Implement this function.

Definition at line 153 of file regex.h.

4.950.3.8 template<typename _Ch_type > char_type **std::regex_traits**< _Ch_type >::translate (char_type *__c*) const [inline]

Performs the identity translation.

Parameters

__c	A character to the locale-specific character set.

Returns

__c.

Definition at line 87 of file regex.h.

4.950.3.9 template<typename _Ch_type > char_type **std::regex_traits**< _Ch_type >::translate_nocase (char_type *__c*) const [inline]

Translates a character into a case-insensitive equivalent.

Parameters

__c	A character to the locale-specific character set.

Returns

the locale-specific lower-case equivalent of __c.

Exceptions

std::bad_cast	if the imbued locale does not support the ctype facet.

Definition at line 100 of file regex.h.

The documentation for this struct was generated from the following file:

- regex.h

4.951 std::remove_all_extents< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

4.951.1 Detailed Description

template<typename _Tp>struct std::remove_all_extents< _Tp >

remove_all_extents

Definition at line 1637 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.952 std::remove_const< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

4.952.1 Detailed Description

template<typename _Tp>struct std::remove_const< _Tp >

remove_const

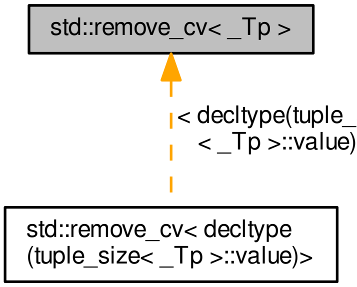
Definition at line 1327 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.953 std::remove_cv< _Tp > Struct Template Reference

Inheritance diagram for std::remove_cv< _Tp >:



Public Types

- typedef remove_const< typename remove_volatile< _Tp >::type >::type **type**

4.953.1 Detailed Description

template<typename _Tp>struct std::remove_cv< _Tp >

remove_cv

Definition at line 1345 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.954 std::remove_extent< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

4.954.1 Detailed Description

template<typename _Tp>struct std::remove_extent< _Tp >

remove_extent

Definition at line 1624 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.955 std::remove_pointer< _Tp > Struct Template Reference

Inherits std::__remove_pointer_helper< _Tp, typename >.

Public Types

- typedef _Tp **type**

4.955.1 Detailed Description

template<typename _Tp>struct std::remove_pointer< _Tp >

remove_pointer

Definition at line 1661 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.956 std::remove_reference< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

4.956.1 Detailed Description

template<typename _Tp>struct std::remove_reference< _Tp >

remove_reference

Definition at line 1374 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.957 std::remove_volatile< _Tp > Struct Template Reference

Public Types

- typedef _Tp **type**

4.957.1 Detailed Description

template<typename _Tp>struct std::remove_volatile< _Tp >

remove_volatile

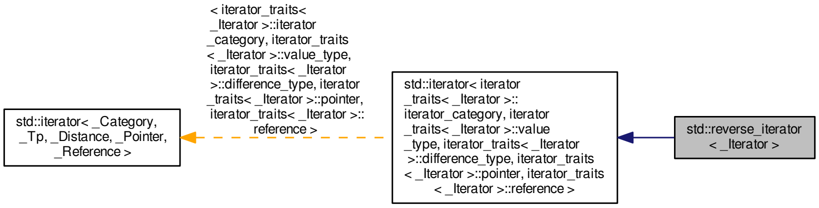
Definition at line 1336 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.958 std::reverse_iterator< _Iterator > Class Template Reference

Inheritance diagram for std::reverse_iterator< _Iterator >:



Public Types

- typedef __traits_type::difference_type **difference_type**
- typedef iterator_traits < _Iterator > ::iterator_category iterator_category

- typedef _Iterator **iterator_type**
- typedef __traits_type::pointer **pointer**
- typedef __traits_type::reference **reference**
- typedef iterator_traits
  < _Iterator >::value_type value_type

Public Member Functions

- reverse_iterator ()
- reverse_iterator (iterator_type __x)
- reverse_iterator (const reverse_iterator &__x)
- template<typename _Iter >
  reverse_iterator (const reverse_iterator< _Iter > &__x)
- iterator_type base () const
- reference operator* () const
- reverse_iterator operator+ (difference_type __n) const
- reverse_iterator & operator++ ()
- reverse_iterator operator++ (int)
- reverse_iterator & operator+= (difference_type __n)
- reverse_iterator operator- (difference_type __n) const
- reverse_iterator & operator-- ()
- reverse_iterator operator-- (int)
- reverse_iterator & operator-= (difference_type __n)
- pointer operator-> () const
- reference operator[] (difference_type __n) const

Protected Types

- typedef iterator_traits
  < _Iterator > **__traits_type**

Protected Attributes

- _Iterator **current**

4.958.1 Detailed Description

template<typename _Iterator>class std::reverse_iterator< _Iterator >

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 96 of file stl_iterator.h.

4.958.2 Member Typedef Documentation

4.958.2.1 typedef iterator_traits< _Iterator >::iterator_category std::iterator< iterator_traits< _Iterator >::iterator_category , iterator_traits< _Iterator >::value_type , iterator_traits< _Iterator >::difference_type , iterator_traits< _Iterator >::pointer , iterator_traits< _Iterator >::reference >::iterator_category `[inherited]`

One of the tag types.

Definition at line 121 of file stl_iterator_base_types.h.

4.958.2.2 typedef iterator_traits< _Iterator >::value_type std::iterator< iterator_traits< _Iterator >::iterator_category , iterator_traits< _Iterator >::value_type , iterator_traits< _Iterator >::difference_type , iterator_traits< _Iterator >::pointer , iterator_traits< _Iterator >::reference >::value_type `[inherited]`

The type "pointed to" by the iterator.

Definition at line 123 of file stl_iterator_base_types.h.

4.958.3 Constructor & Destructor Documentation

4.958.3.1 template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator ( ) `[inline]`

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 120 of file stl_iterator.h.

Referenced by std::reverse_iterator< _Iterator >::operator+(), and std::reverse_iterator< _Iterator >::operator-().

4.958.3.2 template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator (iterator_type *__x*) `[inline]`, `[explicit]`

This iterator will move in the opposite direction that `x` does.

Definition at line 126 of file stl_iterator.h.

4.958.3.3 template<typename _Iterator> std::reverse_iterator< _Iterator >::reverse_iterator (const reverse_iterator< _Iterator > & *__x*) `[inline]`

The copy constructor is normal.

Definition at line 131 of file stl_iterator.h.

4.958.3.4 template<typename _Iterator> template<typename _Iter > std::reverse_iterator< _Iterator >::reverse_iterator (const reverse_iterator< _Iter > & *__x*) `[inline]`

A reverse_iterator across other types can be copied if the underlying iterator can be converted to the type of `current`.

Definition at line 139 of file stl_iterator.h.

4.958.4 Member Function Documentation

4.958.4.1 template<typename _Iterator> iterator_type std::reverse_iterator< _Iterator >::base ( ) const `[inline]`

Returns

`current`, the iterator used for underlying work.

Definition at line 146 of file stl_iterator.h.

Referenced by std::operator==().

4.958.4.2 template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator* ( ) const [inline]

Returns

A reference to the value at `--current`

This requires that `--current` is dereferenceable.

Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: http://gcc.gnu.org/PR51823

Definition at line 160 of file stl_iterator.h.

Referenced by std::reverse_iterator< _Iterator >::operator->().

4.958.4.3 template<typename _Iterator> reverse_iterator std::reverse_iterator< _Iterator >::operator+ (difference_type __n) const [inline]

Returns

A reverse_iterator that refers to `current` - *__n*

The underlying iterator must be a Random Access Iterator.

Definition at line 231 of file stl_iterator.h.

References std::reverse_iterator< _Iterator >::reverse_iterator().

4.958.4.4 template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator++ ( ) [inline]

Returns

`*this`

Decrements the underlying iterator.

Definition at line 181 of file stl_iterator.h.

4.958.4.5 template<typename _Iterator> reverse_iterator std::reverse_iterator< _Iterator >::operator++ (int) [inline]

Returns

The original value of `*this`

Decrements the underlying iterator.

Definition at line 193 of file stl_iterator.h.

4.958.4.6 template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator+= (difference_type __n) [inline]

Returns

*this

Moves the underlying iterator backwards *__n* steps. The underlying iterator must be a Random Access Iterator.

Definition at line 241 of file stl_iterator.h.

4.958.4.7 template<typename _Iterator> reverse_iterator std::reverse_iterator< _Iterator >::operator- (difference_type __n) const [inline]

Returns

A reverse_iterator that refers to `current` - *__n*

The underlying iterator must be a Random Access Iterator.

Definition at line 253 of file stl_iterator.h.

References std::reverse_iterator< _Iterator >::reverse_iterator().

4.958.4.8 template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-- ( ) [inline]

Returns

`*this`

Increments the underlying iterator.

Definition at line 206 of file stl_iterator.h.

4.958.4.9 template<typename _Iterator> reverse_iterator std::reverse_iterator< _Iterator >::operator-- (int) [inline]

Returns

A reverse_iterator with the previous value of `*this`

Increments the underlying iterator.

Definition at line 218 of file stl_iterator.h.

4.958.4.10 template<typename _Iterator> reverse_iterator& std::reverse_iterator< _Iterator >::operator-= (difference_type __n) [inline]

Returns

*this

Moves the underlying iterator forwards *__n* steps. The underlying iterator must be a Random Access Iterator.

Definition at line 263 of file stl_iterator.h.

4.958.4.11 template<typename _Iterator> pointer std::reverse_iterator< _Iterator >::operator-> ( ) const [inline]

Returns

A pointer to the value at `--current`

This requires that `--current` is dereferenceable.

Definition at line 172 of file stl_iterator.h.

References std::reverse_iterator< _Iterator >::operator*().

4.958.4.12 template<typename _Iterator> reference std::reverse_iterator< _Iterator >::operator[] (difference_type __n) const [inline]

Returns

The value at `current` - __*n* - 1

The underlying iterator must be a Random Access Iterator.

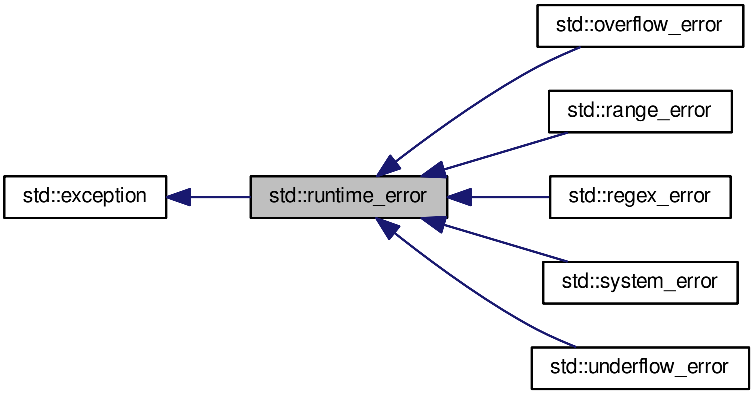
Definition at line 275 of file stl_iterator.h.

The documentation for this class was generated from the following file:

- stl_iterator.h

4.959 std::runtime_error Class Reference

Inheritance diagram for std::runtime_error:



Public Member Functions

- runtime_error (const string &__arg)
- virtual const char ∗ what () const noexcept

4.959.1 Detailed Description

One of two subclasses of exception.

Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 112 of file stdexcept.

4.959.2 Constructor & Destructor Documentation

4.959.2.1 std::runtime_error::runtime_error (const string & *__arg*) `[explicit]`

Takes a character string describing the error.

4.959.3 Member Function Documentation

4.959.3.1 virtual const char∗ std::runtime_error::what ( ) const [virtual], [noexcept]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

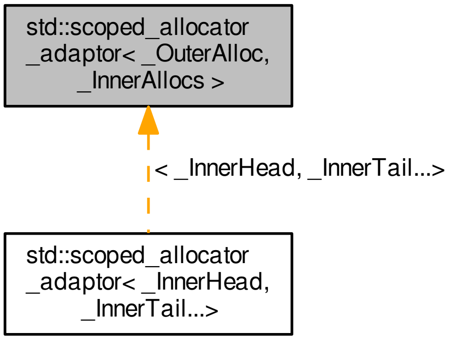
Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- stdexcept

4.960 std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference

Inheritance diagram for std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >:



Public Types

- typedef __traits::const_pointer **const_pointer**
- typedef __traits::const_void_pointer **const_void_pointer**
- typedef __traits::difference_type **difference_type**
- typedef __inner_type::__type **inner_allocator_type**
- typedef _OuterAlloc **outer_allocator_type**
- typedef __traits::pointer **pointer**
- typedef conditional< __any_of < __propagate_on_copy, _OuterAlloc, _InnerAllocs...> ::value, true_type, false_type > ::type **propagate_on_container_copy_assignment**
- typedef conditional< __any_of < __propagate_on_move, _OuterAlloc, _InnerAllocs...> ::value, true_type, false_type > ::type **propagate_on_container_move_assignment**

- typedef conditional< __any_of
  < __propagate_on_swap,
  _OuterAlloc, _InnerAllocs...>
  ::value, true_type, false_type >
  ::type **propagate_on_container_swap**
- typedef __traits::size_type **size_type**
- typedef __traits::value_type **value_type**
- typedef __traits::void_pointer **void_pointer**

Public Member Functions

- template<typename _Outer2 >
  scoped_allocator_adaptor (_Outer2 &&__outer, const _InnerAllocs &...__inner)
- **scoped_allocator_adaptor** (const scoped_allocator_adaptor &__other)
- **scoped_allocator_adaptor** (scoped_allocator_adaptor &&__other)
- template<typename _Outer2 >
  scoped_allocator_adaptor (const scoped_allocator_adaptor< _Outer2, _InnerAllocs...> &__other)
- template<typename _Outer2 >
  scoped_allocator_adaptor (scoped_allocator_adaptor< _Outer2, _InnerAllocs...> &&__other)
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const_void_pointer __hint)
- template<typename _Tp , typename... _Args>
  void **construct** (_Tp ∗__p, _Args &&...__args)
- template<typename _T1 , typename _T2 , typename... _Args1, typename... _Args2>
  void **construct** (pair< _T1, _T2 > ∗__p, piecewise_construct_t, tuple< _Args1...> __x, tuple< _Args2...> __y)
- template<typename _T1 , typename _T2 >
  void **construct** (pair< _T1, _T2 > ∗__p)
- template<typename _T1 , typename _T2 , typename _Up , typename _Vp >
  void **construct** (pair< _T1, _T2 > ∗__p, _Up &&__u, _Vp &&__v)
- template<typename _T1 , typename _T2 , typename _Up , typename _Vp >
  void **construct** (pair< _T1, _T2 > ∗__p, const pair< _Up, _Vp > &__x)
- template<typename _T1 , typename _T2 , typename _Up , typename _Vp >
  void **construct** (pair< _T1, _T2 > ∗__p, pair< _Up, _Vp > &&__x)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Tp >
  void **destroy** (_Tp ∗__p)
- inner_allocator_type & **inner_allocator** () noexcept
- const inner_allocator_type & **inner_allocator** () const noexcept
- size_type **max_size** () const
- outer_allocator_type & **outer_allocator** () noexcept
- const outer_allocator_type & **outer_allocator** () const noexcept
- scoped_allocator_adaptor **select_on_container_copy_construction** () const

Friends

- template<typename _Outer , typename... _Inner>
  class **scoped_allocator_adaptor**
- template<typename... >
  class **__inner_type_impl**
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept

4.960.1 Detailed Description

template<typename _OuterAlloc, typename... _InnerAllocs>class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >

Primary class template.

Definition at line 176 of file scoped_allocator.

The documentation for this class was generated from the following file:

- scoped_allocator

4.961 std::seed_seq Class Reference

Public Types

- typedef uint_least32_t result_type

Public Member Functions

- seed_seq ()
- template<typename _IntType >
  seed_seq (std::initializer_list< _IntType > il)
- template<typename _InputIterator >
  seed_seq (_InputIterator __begin, _InputIterator __end)
- template<typename _RandomAccessIterator >
  void **generate** (_RandomAccessIterator __begin, _RandomAccessIterator __end)
- template<typename OutputIterator >
  void **param** (OutputIterator __dest) const
- size_t **size** () const

4.961.1 Detailed Description

The seed_seq class generates sequences of seeds for random number generators.

Definition at line 6024 of file random.h.

4.961.2 Member Typedef Documentation

4.961.2.1 typedef uint_least32_t **std::seed_seq::result_type**

The type of the seed vales.

Definition at line 6029 of file random.h.

4.961.3 Constructor & Destructor Documentation

4.961.3.1 std::seed_seq::seed_seq ( ) `[inline]`

Default constructor.

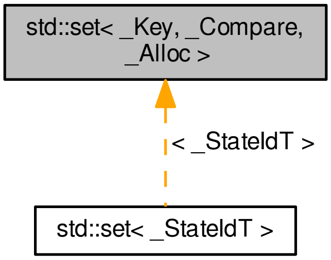
Definition at line 6032 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.962 std::set< _Key, _Compare, _Alloc > Class Template Reference

Inheritance diagram for std::set< _Key, _Compare, _Alloc >:



Public Types

- typedef _Key key_type
- typedef _Key value_type
- typedef _Compare key_compare
- typedef _Compare value_compare
- typedef _Alloc allocator_type

- typedef _Key_alloc_type::pointer pointer
- typedef _Key_alloc_type::const_pointer const_pointer
- typedef _Key_alloc_type::reference reference
- typedef _Key_alloc_type::const_reference const_reference
- typedef _Rep_type::const_iterator iterator
- typedef _Rep_type::const_iterator const_iterator
- typedef _Rep_type::const_reverse_iterator reverse_iterator
- typedef _Rep_type::const_reverse_iterator const_reverse_iterator
- typedef _Rep_type::size_type size_type
- typedef _Rep_type::difference_type difference_type

Public Member Functions

- set ()
- set (const _Compare &__comp, const allocator_type &__a=allocator_type())

- template<typename _InputIterator >
  set (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator >
  set (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- set (const set &__x)
- set (set &&__x) noexcept(is_nothrow_copy_constructible< _Compare >
- set (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- iterator begin () const noexcept
- iterator cbegin () const noexcept
- iterator cend () const noexcept
- void clear () noexcept
- size_type count (const key_type &__x) const
- reverse_iterator crbegin () const noexcept
- reverse_iterator crend () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- iterator end () const noexcept
- iterator erase (const_iterator __position)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- iterator insert (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_compare key_comp () const
- size_type max_size () const noexcept
- set & operator= (const set &__x)
- set & operator= (set &&__x)
- set & operator= (initializer_list< value_type > __l)
- reverse_iterator rbegin () const noexcept
- reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (set &__x)
- value_compare value_comp () const

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const

- iterator upper_bound (const key_type &__x)

- const_iterator upper_bound (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > equal_range (const key_type &__x) const

Friends

- template<typename _K1 , typename _C1 , typename _A1 >
  bool **operator<** (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)
- template<typename _K1 , typename _C1 , typename _A1 >
  bool **operator==** (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)

4.962.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>class std::set< _Key, _Compare, _Alloc >

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Template Parameters

_Key	Type of key objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a `container`, a `reversible container`, and an `associative container` (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (∗_unique versus ∗_equal, same as the standard).

Definition at line 90 of file stl_set.h.

4.962.2 Member Typedef Documentation

4.962.2.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Alloc **std::set< _Key, _Compare, _Alloc >::allocator_type**

Public typedefs.

Definition at line 107 of file stl_set.h.

4.962.2.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_iterator **std::set< _Key, _Compare, _Alloc >::const_iterator**

Iterator-related typedefs.

Definition at line 128 of file stl_set.h.

4.962.2.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::const_pointer std::set< _Key, _Compare, _Alloc >::const_pointer

Iterator-related typedefs.

Definition at line 121 of file stl_set.h.

4.962.2.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::const_reference std::set< _Key, _Compare, _Alloc >::const_reference

Iterator-related typedefs.

Definition at line 123 of file stl_set.h.

4.962.2.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_iterator

Iterator-related typedefs.

Definition at line 130 of file stl_set.h.

4.962.2.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type

Iterator-related typedefs.

Definition at line 132 of file stl_set.h.

4.962.2.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator

Iterator-related typedefs.

Definition at line 127 of file stl_set.h.

4.962.2.8 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare

Public typedefs.

Definition at line 105 of file stl_set.h.

4.962.2.9 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key std::set< _Key, _Compare, _Alloc >::key_type

Public typedefs.

Definition at line 103 of file stl_set.h.

4.962.2.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::pointer std::set< _Key, _Compare, _Alloc >::pointer

Iterator-related typedefs.

Definition at line 120 of file stl_set.h.

4.962.2.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key_alloc_type::reference std::set< _Key, _Compare, _Alloc >::reference

Iterator-related typedefs.

Definition at line 122 of file stl_set.h.

4.962.2.12 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator

Iterator-related typedefs.

Definition at line 129 of file stl_set.h.

4.962.2.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Rep_type::size_type std::set< _Key, _Compare, _Alloc >::size_type

Iterator-related typedefs.

Definition at line 131 of file stl_set.h.

4.962.2.14 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Compare std::set< _Key, _Compare, _Alloc >::value_compare

Public typedefs.

Definition at line 106 of file stl_set.h.

4.962.2.15 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> typedef _Key std::set< _Key, _Compare, _Alloc >::value_type

Public typedefs.

Definition at line 104 of file stl_set.h.

4.962.3 Constructor & Destructor Documentation

4.962.3.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::set< _Key, _Compare, _Alloc >::set ( ) `[inline]`

Default constructor creates no elements.

Definition at line 139 of file stl_set.h.

4.962.3.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::set< _Key, _Compare, _Alloc >::set (const _Compare & *__comp*, const allocator_type & *__a* = allocator_type`()`) `[inline]`, `[explicit]`

Creates a set with no elements.

Parameters

__comp	Comparator to use.
__a	An allocator object.

Definition at line 148 of file stl_set.h.

4.962.3.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > std::set< _Key, _Compare, _Alloc >::set (_InputIterator *__first*, _InputIterator *__last*) `[inline]`

Builds a set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a set consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 163 of file stl_set.h.

4.962.3.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > std::set< _Key, _Compare, _Alloc >::set (_InputIterator *__first*, _InputIterator *__last*, const _Compare & *__comp*, const allocator_type & *__a* = allocator_type()) [inline]

Builds a set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a set consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

Definition at line 180 of file stl_set.h.

4.962.3.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::set< _Key, _Compare, _Alloc >::set (const set< _Key, _Compare, _Alloc > & *__x*) [inline]

Set copy constructor.

Parameters

__x	A set of identical element and allocator types.

The newly-created set uses a copy of the allocation object used by __*x*.

Definition at line 193 of file stl_set.h.

4.962.3.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::set< _Key, _Compare, _Alloc >::set (set< _Key, _Compare, _Alloc > && *__x*) [inline], [noexcept]

Set move constructor

Parameters

__x	A set of identical element and allocator types.

The newly-created set contains the exact contents of *x*. The contents of *x* are a valid, but unspecified set.

Definition at line 204 of file stl_set.h.

4.962.3.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::set< _Key, _Compare, _Alloc >::set (initializer_list< value_type > *__l*, const _Compare & *__comp* = _Compare(), const allocator_type & *__a* = allocator_type()) [inline]

Builds a set from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

Definition at line 218 of file stl_set.h.

4.962.4 Member Function Documentation

4.962.4.1 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 298 of file stl_set.h.

4.962.4.2 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 335 of file stl_set.h.

4.962.4.3 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 344 of file stl_set.h.

4.962.4.4 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::set< _Key, _Compare, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 628 of file stl_set.h.

Referenced by std::set< _StateIdT >::operator=().

4.962.4.5 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::set< _Key, _Compare, _Alloc >::count (const key_type & *__x*) const [inline]

Finds the number of elements.

Parameters

__x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 642 of file stl_set.h.

4.962.4.6 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set< _Key, _Compare, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 353 of file stl_set.h.

4.962.4.7 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set< _Key, _Compare, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 362 of file stl_set.h.

4.962.4.8 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename... _Args> std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::emplace (_Args &&... *__args*) [inline]

Attempts to build and insert an element into the set.

Parameters

__args	Arguments used to generate an element.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 413 of file stl_set.h.

4.962.4.9 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename... _Args> iterator std::set< _Key, _Compare, _Alloc >::emplace_hint (const_iterator *__pos*, _Args &&... *__args*) [inline]

Attempts to insert an element into the set.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from *__args* (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 439 of file stl_set.h.

4.962.4.10 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> bool std::set< _Key, _Compare, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the set is empty.

Definition at line 368 of file stl_set.h.

4.962.4.11 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 307 of file stl_set.h.

4.962.4.12 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::pair<iterator, iterator> std::set< _Key, _Compare, _Alloc >::equal_range (const key_type & __x) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 722 of file stl_set.h.

4.962.4.13 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::pair<const_iterator, const_iterator> std::set< _Key, _Compare, _Alloc >::equal_range (const key_type & __x) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 726 of file stl_set.h.

4.962.4.14 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::erase (const_iterator *__position*) [inline]

Erases an element from a set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 550 of file stl_set.h.

4.962.4.15 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::set< _Key, _Compare, _Alloc >::erase (const key_type & *__x*) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 580 of file stl_set.h.

4.962.4.16 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::erase (const_iterator *__first*, const_iterator *__last*) [inline]

Erases a [__first,__last) range of elements from a set.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 601 of file stl_set.h.

4.962.4.17 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::find (const key_type & *__x*) [inline]

Tries to locate an element in a set.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 660 of file stl_set.h.

4.962.4.18 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> const_iterator std::set< _Key, _Compare, _Alloc >::find (const key_type & *__x*) const [inline]

Tries to locate an element in a set.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 664 of file stl_set.h.

4.962.4.19 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Returns the allocator object with which the set was constructed.

Definition at line 289 of file stl_set.h.

4.962.4.20 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::insert (const value_type & __x) [inline]

Attempts to insert an element into the set.

Parameters

__x	Element to be inserted.

Returns

> A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 460 of file stl_set.h.

Referenced by std::set< _StateIdT >::operator=().

4.962.4.21 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::insert (const_iterator __position, const value_type & __x) [inline]

Attempts to insert an element into the set.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

> An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 497 of file stl_set.h.

4.962.4.22 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> template<typename _InputIterator > void std::set< _Key, _Compare, _Alloc >::insert (_InputIterator __first, _InputIterator __last) [inline]

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 517 of file stl_set.h.

4.962.4.23 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::set< _Key, _Compare, _Alloc >::insert (initializer_list< value_type > __l) [inline]

Attempts to insert a list of elements into the set.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 529 of file stl_set.h.

Referenced by std::set< _StateIdT >::insert().

4.962.4.24 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> key_compare std::set< _Key, _Compare, _Alloc >::key_comp ( ) const [inline]

Returns the comparison object with which the set was constructed.

Definition at line 281 of file stl_set.h.

4.962.4.25 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::lower_bound (const key_type & __x) [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 681 of file stl_set.h.

4.962.4.26 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound (const key_type & __x) const [inline]

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 685 of file stl_set.h.

4.962.4.27 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::set< _Key, _Compare, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the set.

Definition at line 378 of file stl_set.h.

4.962.4.28 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set& std::set< _Key, _Compare, _Alloc >::operator= (const set< _Key, _Compare, _Alloc > & __x) [inline]

Set assignment operator.

Parameters

__x	A set of identical element and allocator types.

All the elements of *__x* are copied, but unlike the copy constructor, the allocator object is not copied.

Definition at line 233 of file stl_set.h.

4.962.4.29 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set& std::set< _Key, _Compare, _Alloc >::operator= (set< _Key, _Compare, _Alloc > && __x) [inline]

Set move assignment operator.

Parameters

__x	A set of identical element and allocator types.

The contents of *__x* are moved into this set (without copying). *__x* is a valid, but unspecified set.

Definition at line 248 of file stl_set.h.

4.962.4.30 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> set& std::set< _Key, _Compare, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Set list assignment operator.

Parameters

__l	An initializer_list.

This function fills a set with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 269 of file stl_set.h.

4.962.4.31 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 316 of file stl_set.h.

4.962.4.32 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> reverse_iterator std::set< _Key, _Compare, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 325 of file stl_set.h.

4.962.4.33 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> size_type std::set< _Key, _Compare, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the set.

Definition at line 373 of file stl_set.h.

4.962.4.34 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> void std::set< _Key, _Compare, _Alloc >::swap (set< _Key, _Compare, _Alloc > & __x) [inline]

Swaps data with another set.

Parameters

__x	A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

Definition at line 393 of file stl_set.h.

Referenced by std::set< _StateIdT >::operator=(), and std::swap().

4.962.4.35 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> iterator std::set< _Key, _Compare, _Alloc >::upper_bound (const key_type & __x) [inline]

Finds the end of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 697 of file stl_set.h.

4.962.4.36 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> const_iterator std::set< _Key, _Compare, _Alloc >::upper_bound (const key_type & __x) const [inline]

Finds the end of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 701 of file stl_set.h.

4.962.4.37 template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> value_compare std::set< _Key, _Compare, _Alloc >::value_comp ( ) const [inline]

Returns the comparison object with which the set was constructed.

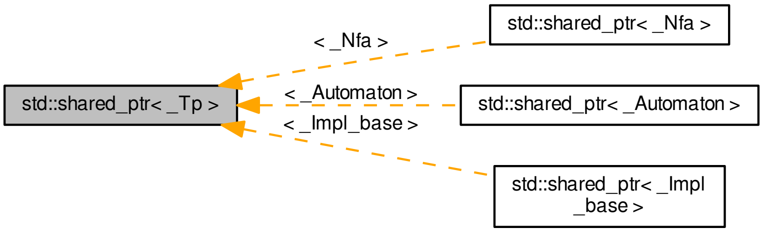
Definition at line 285 of file stl_set.h.

The documentation for this class was generated from the following file:

- stl_set.h

4.963 std::shared_ptr< _Tp > Class Template Reference

Inheritance diagram for std::shared_ptr< _Tp >:



Public Types

- typedef _Tp **element_type**

Public Member Functions

- constexpr shared_ptr () noexcept
- **shared_ptr** (const shared_ptr &) noexcept=default
- template<typename _Tp1 >
  shared_ptr (_Tp1 ∗__p)
- template<typename _Tp1 , typename _Deleter >
  shared_ptr (_Tp1 ∗__p, _Deleter __d)
- template<typename _Deleter >
  shared_ptr (nullptr_t __p, _Deleter __d)

- template<typename _Tp1 , typename _Deleter , typename _Alloc >
  shared_ptr (_Tp1 ∗__p, _Deleter __d, _Alloc __a)
- template<typename _Deleter , typename _Alloc >
  shared_ptr (nullptr_t __p, _Deleter __d, _Alloc __a)
- template<typename _Tp1 >
  shared_ptr (const shared_ptr< _Tp1 > &__r, _Tp ∗__p) noexcept
- template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1∗, _Tp∗>::value>::type>
  shared_ptr (const shared_ptr< _Tp1 > &__r) noexcept
- shared_ptr (shared_ptr &&__r) noexcept
- template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1∗, _Tp∗>::value>::type>
  shared_ptr (shared_ptr< _Tp1 > &&__r) noexcept
- template<typename _Tp1 >
  shared_ptr (const weak_ptr< _Tp1 > &__r)
- template<typename _Tp1 , typename _Del >
  shared_ptr (std::unique_ptr< _Tp1, _Del > &&__r)
- constexpr shared_ptr (nullptr_t __p) noexcept
- template<typename _Tp1 >
  shared_ptr (std::auto_ptr< _Tp1 > &&__r)
- _Tp ∗ **get** () const noexcept
- **operator bool** () const
- std::add_lvalue_reference< _Tp >
  ::type **operator∗** () const noexcept
- _Tp ∗ **operator->** () const noexcept
- shared_ptr & **operator=** (const shared_ptr &) noexcept=default
- template<typename _Tp1 >
  shared_ptr & **operator=** (const shared_ptr< _Tp1 > &__r) noexcept
- shared_ptr & **operator=** (shared_ptr &&__r) noexcept
- template<class _Tp1 >
  shared_ptr & **operator=** (shared_ptr< _Tp1 > &&__r) noexcept
- template<typename _Tp1 , typename _Del >
  shared_ptr & **operator=** (std::unique_ptr< _Tp1, _Del > &&__r)
- template<typename _Tp1 >
  bool **owner_before** (__shared_ptr< _Tp1, _Lp > const &__rhs) const
- template<typename _Tp1 >
  bool **owner_before** (__weak_ptr< _Tp1, _Lp > const &__rhs) const
- void **reset** () noexcept
- template<typename _Tp1 >
  void **reset** (_Tp1 ∗__p)
- template<typename _Tp1 , typename _Deleter >
  void **reset** (_Tp1 ∗__p, _Deleter __d)
- template<typename _Tp1 , typename _Deleter , typename _Alloc >
  void **reset** (_Tp1 ∗__p, _Deleter __d, _Alloc __a)
- void **swap** (__shared_ptr< _Tp, _Lp > &__other) noexcept
- bool **unique** () const noexcept
- long **use_count** () const noexcept

Friends

- template<typename _Tp1 , typename _Alloc , typename... _Args>
  shared_ptr< _Tp1 > allocate_shared (const _Alloc &__a, _Args &&...__args)

4.963.1 Detailed Description

template<typename _Tp>class std::shared_ptr< _Tp >

A smart pointer with reference-counted copy semantics.

The object pointed to is deleted when the last shared_ptr pointing to it is destroyed or reset.

Definition at line 93 of file shared_ptr.h.

4.963.2 Constructor & Destructor Documentation

4.963.2.1 template<typename _Tp> constexpr std::shared_ptr< _Tp >::shared_ptr ( ) `[inline]`, `[noexcept]`

Construct an empty shared_ptr.

Postcondition

use_count()==0 && get()==0

Definition at line 100 of file shared_ptr.h.

4.963.2.2 template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p) `[inline]`, `[explicit]`

Construct a shared_ptr that owns the pointer *__p*.

Parameters

__p	A pointer that is convertible to element_type*.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case `delete` *__p* is called.

Definition at line 112 of file shared_ptr.h.

4.963.2.3 template<typename _Tp> template<typename _Tp1 , typename _Deleter > std::shared_ptr< _Tp >::shared_ptr (_Tp1 * __p, _Deleter __d) `[inline]`

Construct a shared_ptr that owns the pointer *__p* and the deleter *__d*.

Parameters

__p	A pointer.
__d	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw

__shared_ptr will release __p by calling __d(__p)

Definition at line 129 of file shared_ptr.h.

4.963.2.4 template<typename _Tp> template<typename _Deleter > std::shared_ptr< _Tp >::shared_ptr (nullptr_t *__p*, _Deleter *__d*) `[inline]`

Construct a shared_ptr that owns a null pointer and the deleter *__d*.

Parameters

__p	A null pointer constant.
__d	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw

The last owner will call __d(__p)

Definition at line 146 of file shared_ptr.h.

4.963.2.5 template<typename _Tp> template<typename _Tp1 , typename _Deleter , typename _Alloc > std::shared_ptr< _Tp >::shared_ptr (_Tp1 ∗ *__p*, _Deleter *__d*, _Alloc *__a*) `[inline]`

Construct a shared_ptr that owns the pointer *__p* and the deleter *__d*.

Parameters

__p	A pointer.
__d	A deleter.
__a	An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

__shared_ptr will release __p by calling __d(__p)

Definition at line 165 of file shared_ptr.h.

4.963.2.6 template<typename _Tp> template<typename _Deleter , typename _Alloc > std::shared_ptr< _Tp >::shared_ptr (nullptr_t *__p*, _Deleter *__d*, _Alloc *__a*) `[inline]`

Construct a shared_ptr that owns a null pointer and the deleter *__d*.

Parameters

__p	A null pointer constant.
__d	A deleter.
__a	An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.

The last owner will call __d(__p)

Definition at line 184 of file shared_ptr.h.

4.963.2.7 template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr (const shared_ptr< _Tp1 > & *__r*, _Tp ∗ *__p*) `[inline]`, `[noexcept]`

Constructs a shared_ptr instance that stores *__p* and shares ownership with *__r*.

Parameters

__r	A shared_ptr.
__p	A pointer that will remain valid while ∗*__r* is valid.

Postcondition

get() == __p && use_count() == __r.use_count()

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`.

```
shared_ptr< pair<int,int> > pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 206 of file shared_ptr.h.

4.963.2.8 template<typename _Tp> template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1∗, _Tp∗>::value>::type> std::shared_ptr< _Tp >::shared_ptr (const shared_ptr< _Tp1 > & *__r*) `[inline]`, `[noexcept]`

If *__r* is empty, constructs an empty shared_ptr; otherwise construct a shared_ptr that shares ownership with *__r*.

Parameters

__r	A shared_ptr.

Postcondition

get() == __r.get() && use_count() == __r.use_count()

Definition at line 218 of file shared_ptr.h.

4.963.2.9 template<typename _Tp> std::shared_ptr< _Tp >::shared_ptr (shared_ptr< _Tp > && *__r*) [inline], [noexcept]

Move-constructs a shared_ptr instance from __*r*.

Parameters

__r	A shared_ptr rvalue.

Postcondition

∗this contains the old value of __*r*, __*r* is empty.

Definition at line 226 of file shared_ptr.h.

4.963.2.10 template<typename _Tp> template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1∗, _Tp∗>::value>::type> std::shared_ptr< _Tp >::shared_ptr (shared_ptr< _Tp1 > && *__r*) [inline], [noexcept]

Move-constructs a shared_ptr instance from __*r*.

Parameters

__r	A shared_ptr rvalue.

Postcondition

∗this contains the old value of __*r*, __*r* is empty.

Definition at line 236 of file shared_ptr.h.

4.963.2.11 template<typename _Tp> template<typename _Tp1 > std::shared_ptr< _Tp >::shared_ptr (const weak_ptr< _Tp1 > & *__r*) [inline], [explicit]

Constructs a shared_ptr that shares ownership with __*r* and stores a copy of the pointer stored in __*r*.

Parameters

__r	A weak_ptr.

Postcondition

use_count() == __r.use_count()

Exceptions

bad_weak_ptr	when __r.expired(), in which case the constructor has no effect.

Definition at line 248 of file shared_ptr.h.

4.963.2.12 template<typename _Tp> constexpr std::shared_ptr< _Tp >::shared_ptr (nullptr_t *__p*) [inline], [noexcept]

Construct an empty shared_ptr.

Parameters

__p	A null pointer constant.

Postcondition

use_count() == 0 && get() == nullptr

Definition at line 265 of file shared_ptr.h.

4.963.3 Friends And Related Function Documentation

4.963.3.1 template<typename _Tp> template<typename _Tp1 , typename _Alloc , typename... _Args> shared_ptr<_Tp1> allocate_shared (const _Alloc & *__a*, _Args &&... *__args*) [friend]

Create an object that is owned by a shared_ptr.

Parameters

__a	An allocator.
__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

An	exception thrown from *_Alloc::allocate* or from the constructor of *_Tp*.

A copy of *__a* will be used to allocate memory for the shared_ptr and the new object.

Definition at line 595 of file shared_ptr.h.

The documentation for this class was generated from the following files:

- shared_ptr.h
- auto_ptr.h

4.964 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference

Public Types

- typedef
  _RandomNumberEngine::result_type result_type

Public Member Functions

- shuffle_order_engine ()
- shuffle_order_engine (const _RandomNumberEngine &__rng)
- shuffle_order_engine (_RandomNumberEngine &&__rng)
- shuffle_order_engine (result_type __s)
- template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type>
  shuffle_order_engine (_Sseq &__q)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- void seed (result_type __s)
- template<typename _Sseq >
  void seed (_Sseq &__q)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr size_t **table_size**

Friends

- template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)
- bool operator== (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)
- template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)

4.964.1 Detailed Description

template<typename _RandomNumberEngine, size_t __k>class std::shuffle_order_engine< _RandomNumberEngine, __k >

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1292 of file random.h.

4.964.2 Member Typedef Documentation

4.964.2.1 template<typename _RandomNumberEngine, size_t __k> typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type

The type of the generated random value.

Definition at line 1295 of file random.h.

4.964.3 Constructor & Destructor Documentation

4.964.3.1 template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine ( ) `[inline]`

Constructs a default shuffle_order_engine engine.

The underlying engine is default constructed as well.

Definition at line 1308 of file random.h.

4.964.3.2 template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (const _RandomNumberEngine & *__rng*) `[inline]`, `[explicit]`

Copy constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

Definition at line 1319 of file random.h.

4.964.3.3 template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (_RandomNumberEngine && *__rng*) `[inline]`, `[explicit]`

Move constructs a shuffle_order_engine engine.

Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

Definition at line 1330 of file random.h.

4.964.3.4 template<typename _RandomNumberEngine, size_t __k> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (result_type *__s*) `[inline]`, `[explicit]`

Seed constructs a shuffle_order_engine engine.

Constructs the underlying generator engine seeded with `__s`.

Parameters

__s	A seed value for the base class engine.

Definition at line 1341 of file random.h.

4.964.3.5 template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq , typename = typename std::enable_if<!std::is_same<_Sseq, shuffle_order_engine>::value && !std::is_same<_Sseq, _RandomNumberEngine>::value> ::type> std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (_Sseq & *__q*) `[inline]`, `[explicit]`

Generator construct a shuffle_order_engine engine.

Parameters

__q	A seed sequence.

Definition at line 1355 of file random.h.

4.964.4 Member Function Documentation

4.964.4.1 template<typename _RandomNumberEngine, size_t __k> const _RandomNumberEngine& std::shuffle_order_engine< _RandomNumberEngine, __k >::base ( ) const `[inline]`, `[noexcept]`

Gets a const reference to the underlying generator engine object.

Definition at line 1398 of file random.h.

4.964.4.2 template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard (unsigned long long *__z*) `[inline]`

Discard a sequence of random numbers.

Definition at line 1419 of file random.h.

4.964.4.3 template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::max ( ) `[inline]`, `[static]`

Gets the maximum value in the generated random number range.

Definition at line 1412 of file random.h.

References std::max().

4.964.4.4 template<typename _RandomNumberEngine, size_t __k> static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::min ( ) `[inline]`, `[static]`

Gets the minimum value in the generated random number range.

Definition at line 1405 of file random.h.

References std::min().

4.964.4.5 template<typename _RandomNumberEngine , size_t __k> shuffle_order_engine< _RandomNumberEngine, __k >::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::operator() ( )

Gets the next value in the generated random number sequence.

Definition at line 818 of file bits/random.tcc.

4.964.4.6 template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ( ) `[inline]`

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

Definition at line 1364 of file random.h.

4.964.4.7 template<typename _RandomNumberEngine, size_t __k> void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (result_type __s) [inline]

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

Definition at line 1375 of file random.h.

4.964.4.8 template<typename _RandomNumberEngine, size_t __k> template<typename _Sseq > void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (_Sseq & __q) [inline]

Reseeds the shuffle_order_engine object with the given seed sequence.

Parameters

__q	A seed generator function.

Definition at line 1388 of file random.h.

4.964.5 Friends And Related Function Documentation

4.964.5.1 template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]

Inserts the current state of a shuffle_order_engine random number generator engine `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A shuffle_order_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.964.5.2 template<typename _RandomNumberEngine, size_t __k> bool operator== (const shuffle_order_engine< _RandomNumberEngine, __k > & __lhs, const shuffle_order_engine< _RandomNumberEngine, __k > & __rhs) [friend]

Compares two shuffle_order_engine random number generator objects of the same type for equality.

Parameters

__lhs	A shuffle_order_engine random number generator object.
__rhs	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1443 of file random.h.

4.964.5.3 template<typename _RandomNumberEngine, size_t __k> template<typename _RandomNumberEngine1 , size_t __k1, typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & __is, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]

Extracts the current state of a % subtract_with_carry_engine random number generator engine `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A shuffle_order_engine random number generator engine.

Returns

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.965 std::slice Class Reference

Public Member Functions

- slice ()
- slice (size_t __o, size_t __d, size_t __s)
- size_t size () const
- size_t start () const
- size_t stride () const

4.965.1 Detailed Description

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 59 of file slice_array.h.

The documentation for this class was generated from the following file:

- slice_array.h

4.966 std::slice_array< _Tp > Class Template Reference

Public Types

- typedef _Tp **value_type**

Public Member Functions

- slice_array (const slice_array &)
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- slice_array & operator= (const slice_array &)
- void operator= (const valarray< _Tp > &) const
- void operator= (const _Tp &) const
- template<class _Dom >
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const
- template<class _Dom >
  void **operator|=** (const _Expr< _Dom, _Tp > &) const

Friends

- class **valarray< _Tp >**

4.966.1 Detailed Description

template<typename _Tp>class std::slice_array< _Tp >

Reference to one-dimensional subset of an array.

A slice_array is a reference to the actual elements of an array specified by a slice. The way to get a slice_array is to call operator[](slice) on a valarray. The returned slice_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this slice_array refers to.

Parameters

Tp	Element type.

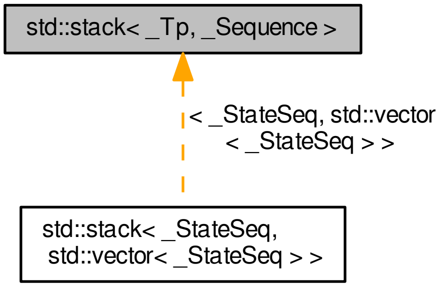
Definition at line 123 of file slice_array.h.

The documentation for this class was generated from the following file:

- slice_array.h

4.967 std::stack< _Tp, _Sequence > Class Template Reference

Inheritance diagram for std::stack< _Tp, _Sequence >:



Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- stack (const _Sequence &__c)

- **stack** (_Sequence &&__c=_Sequence())
- template<typename... _Args>
  void **emplace** (_Args &&...__args)
- bool empty () const
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (stack &__s) noexcept(noexcept(swap(c
- **swap** (c, __s.c)
- reference top ()
- const_reference top () const

Public Attributes

- void __s **c**

Protected Attributes

- _Sequence **c**

Friends

- template<typename _Tp1 , typename _Seq1 >
  bool **operator<** (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)
- template<typename _Tp1 , typename _Seq1 >
  bool **operator==** (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)

4.967.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>>class std::stack< _Tp, _Sequence >

A standard container giving FILO behavior.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to deque<_Tp>.

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `back`, `push_back`, and `pop_front`, such as std::list, std::vector, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 96 of file stl_stack.h.

4.967.2 Constructor & Destructor Documentation

4.967.2.1 template<typename _Tp, typename _Sequence = deque<_Tp>> std::stack< _Tp, _Sequence >::stack (const _Sequence & __c) [inline], [explicit]

Default constructor creates no elements.

Definition at line 134 of file stl_stack.h.

4.967.3 Member Function Documentation

4.967.3.1 template<typename _Tp, typename _Sequence = deque<_Tp>> bool std::stack< _Tp, _Sequence >::empty ( ) const [inline]

Returns true if the stack is empty.

Definition at line 146 of file stl_stack.h.

4.967.3.2 template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack< _Tp, _Sequence >::pop ( ) [inline]

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 212 of file stl_stack.h.

4.967.3.3 template<typename _Tp, typename _Sequence = deque<_Tp>> void std::stack< _Tp, _Sequence >::push (const value_type & __x) [inline]

Add data to the top of the stack.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 186 of file stl_stack.h.

4.967.3.4 template<typename _Tp, typename _Sequence = deque<_Tp>> size_type std::stack< _Tp, _Sequence >::size ( ) const [inline]

Returns the number of elements in the stack.

Definition at line 151 of file stl_stack.h.

4.967.3.5 template<typename _Tp, typename _Sequence = deque<_Tp>> reference std::stack< _Tp, _Sequence >::top ( ) [inline]

Returns a read/write reference to the data at the first element of the stack.

Definition at line 159 of file stl_stack.h.

4.967.3.6 template<typename _Tp, typename _Sequence = deque<_Tp>> const_reference std::stack< _Tp, _Sequence >::top ( ) const [inline]

Returns a read-only (constant) reference to the data at the first element of the stack.

Definition at line 170 of file stl_stack.h.

The documentation for this class was generated from the following file:

- stl_stack.h

4.968 std::student_t_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **student_t_distribution** (_RealType __n=_RealType(1))
- **student_t_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::student_t_distribution< _RealType1 > &__x)

- bool operator== (const student_t_distribution &__d1, const student_t_distribution &__d2)
- template<typename _RealType1 , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::student_t_distribution< _RealType1 > &__x)

4.968.1 Detailed Description

template<typename _RealType = double>class std::student_t_distribution< _RealType >

A student_t_distribution random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt(n\pi)} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} (1 + \frac{x^2}{n})^{-(n+1)/2}$$

Definition at line 3354 of file random.h.

4.968.2 Member Typedef Documentation

4.968.2.1 template<typename _RealType = double> typedef _RealType **std::student_t_distribution**< _RealType >::**result_type**

The type of the range of the distribution.

Definition at line 3357 of file random.h.

4.968.3 Member Function Documentation

4.968.3.1 template<typename _RealType = double> **result_type std::student_t_distribution**< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 3437 of file random.h.

References std::numeric_limits< _Tp >::max().

4.968.3.2 template<typename _RealType = double> **result_type std::student_t_distribution**< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 3430 of file random.h.

References std::numeric_limits< _Tp >::min().

4.968.3.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > **result_type std::student_t_distribution**< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 3445 of file random.h.

References std::sqrt().

4.968.3.4 template<typename _RealType = double> param_type std::student_t_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 3415 of file random.h.

4.968.3.5 template<typename _RealType = double> void std::student_t_distribution< _RealType >::param (const param_type & __param) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 3423 of file random.h.

4.968.3.6 template<typename _RealType = double> void std::student_t_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 3398 of file random.h.

References std::normal_distribution< _RealType >::reset(), and std::gamma_distribution< _RealType >::reset().

4.968.4 Friends And Related Function Documentation

4.968.4.1 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_ostream<_CharT, _Traits>& operator<< (std::basic_ostream< _CharT, _Traits > & __os, const std::student_t_distribution< _RealType1 > & __x) [friend]

Inserts a student_t_distribution random number distribution `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A student_t_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

4.968.4.2 template<typename _RealType = double> bool operator== (const student_t_distribution< _RealType > & __d1, const student_t_distribution< _RealType > & __d2) [friend]

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3494 of file random.h.

4.968.4.3 template<typename _RealType = double> template<typename _RealType1 , typename _CharT , typename _Traits > std::basic_istream<_CharT, _Traits>& operator>> (std::basic_istream< _CharT, _Traits > & __is, std::student_t_distribution< _RealType1 > & __x) [friend]

Extracts a student_t_distribution random number distribution `__x` from the input stream `__is`.

Parameters

__is	An input stream.
__x	A student_t_distribution random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.969 std::student_t_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef student_t_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n=_RealType(1))
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.969.1 Detailed Description

template<typename _RealType = double>struct std::student_t_distribution< _RealType >::param_type

Parameter type.

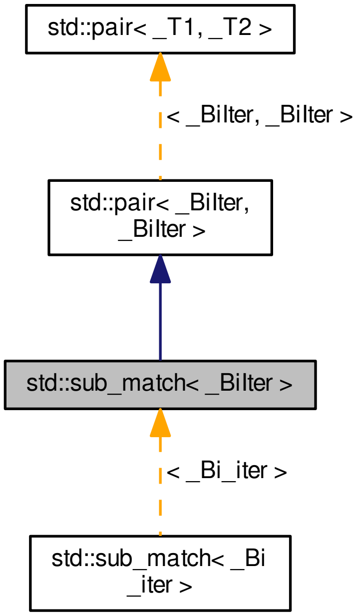
Definition at line 3363 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.970 std::sub_match< _BiIter > Class Template Reference

Inheritance diagram for std::sub_match< _BiIter >:



Public Types

- typedef __iter_traits::difference_type **difference_type**
- typedef _BiIter **first_type**
- typedef _BiIter **iterator**
- typedef _BiIter second_type
- typedef std::basic_string < value_type > **string_type**
- typedef __iter_traits::value_type **value_type**

Public Member Functions

- int compare (const sub_match &__s) const
- int compare (const string_type &__s) const
- int compare (const value_type ∗__s) const
- difference_type length () const
- void __p first && **noexcept** (swap(second, __p.second)))
- operator string_type () const
- string_type str () const
- void **swap** (pair &__p) noexcept(noexcept(swap(first

Public Attributes

- _BIter first
- pair
  is_nothrow_move_assignable
  < _BIter >::value **first**
- bool **matched**
- _BIter second
- **second**
- return ∗ **this**

4.970.1 Detailed Description

template<typename _BiIter>class std::sub_match< _BiIter >

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with std::basic_string objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 741 of file regex.h.

4.970.2 Member Typedef Documentation

4.970.2.1 typedef _BIter std::pair< _BIter , _BIter >::second_type [inherited]

`first_type` is the first bound type

Definition at line 99 of file stl_pair.h.

4.970.3 Member Function Documentation

4.970.3.1 template<typename _BiIter> int std::sub_match< _BiIter >::compare (const sub_match< _BiIter > & __s) const [inline]

Compares this and another matched sequence.

Parameters

__s	Another matched sequence to compare to this one.

Return values

<0	this matched sequence will collate before `__s`.
=0	this matched sequence is equivalent to `__s`.
<0	this matched sequence will collate after `__s`.

Definition at line 802 of file regex.h.

Referenced by std::operator!=(), std::operator<(), std::operator<=(), std::operator==(), std::operator>(), and std::operator>=().

4.970.3.2 template<typename _BiIter> int **std::sub_match**< _BiIter >::compare (const **string_type** & *__s*) const `[inline]`

Compares this sub_match to a string.

Parameters

__s	A string to compare to this sub_match.

Return values

<0	this matched sequence will collate before `__s`.
=0	this matched sequence is equivalent to `__s`.
<0	this matched sequence will collate after `__s`.

Definition at line 815 of file regex.h.

4.970.3.3 template<typename _BiIter> int **std::sub_match**< _BiIter >::compare (const value_type * *__s*) const `[inline]`

Compares this sub_match to a C-style string.

Parameters

__s	A C-style string to compare to this sub_match.

Return values

<0	this matched sequence will collate before `__s`.
=0	this matched sequence is equivalent to `__s`.
<0	this matched sequence will collate after `__s`.

Definition at line 828 of file regex.h.

4.970.3.4 template<typename _BiIter> difference_type **std::sub_match**< _BiIter >::length ( ) const `[inline]`

Gets the length of the matching sequence.

Definition at line 759 of file regex.h.

4.970.3.5 template<typename _BiIter> **std::sub_match**< _BiIter >::operator **string_type** ( ) const `[inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the str() member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 772 of file regex.h.

4.970.3.6 template<typename _BiIter> **string_type** **std::sub_match**< _BiIter >::str ( ) const `[inline]`

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

Definition at line 785 of file regex.h.

Referenced by std::sub_match< _Bi_iter >::compare(), and std::operator<<().

4.970.4 Member Data Documentation

4.970.4.1 _BiIter **std::pair**< _BiIter , _BiIter >::first [inherited]

`second_type` is the second bound type

Definition at line 101 of file stl_pair.h.

Referenced by std::sub_match< _Bi_iter >::length(), std::sub_match< _Bi_iter >::operator string_type(), and std::sub_match< _Bi_iter >::str().

4.970.4.2 _BiIter **std::pair**< _BiIter , _BiIter >::second [inherited]

`first` is a copy of the first object

Definition at line 102 of file stl_pair.h.

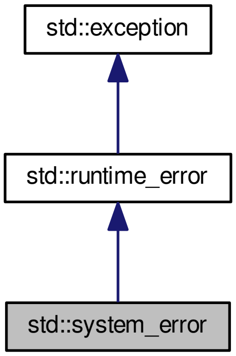
Referenced by std::sub_match< _Bi_iter >::length(), std::sub_match< _Bi_iter >::operator string_type(), and std::sub_match< _Bi_iter >::str().

The documentation for this class was generated from the following file:

- regex.h

4.971 std::system_error Class Reference

Inheritance diagram for std::system_error:



Public Member Functions

- **system_error** (error_code __ec=error_code())
- **system_error** (error_code __ec, const string &__what)
- **system_error** (int __v, const error_category &__ecat)
- **system_error** (int __v, const error_category &__ecat, const string &__what)
- const error_code & **code** () const noexcept
- virtual const char ∗ what () const noexcept

4.971.1 Detailed Description

Thrown to indicate error code of underlying system.

Definition at line 309 of file system_error.

4.971.2 Member Function Documentation

4.971.2.1 virtual const char∗ std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- system_error

4.972 std::thread Class Reference

Classes

- class id

  thread::id

Public Types

- typedef shared_ptr< _Impl_base > **__shared_base_type**
- typedef __gthread_t **native_handle_type**

Public Member Functions

- **thread** (thread &)=delete
- **thread** (const thread &)=delete
- **thread** (thread &&__t) noexcept
- template<typename _Callable , typename... _Args>
  thread (_Callable &&__f, _Args &&...__args)
- void **detach** ()
- thread::id **get_id** () const noexcept
- void **join** ()
- bool **joinable** () const noexcept
- native_handle_type native_handle ()

- thread & **operator=** (const thread &)=delete
- thread & **operator=** (thread &&__t) noexcept
- void **swap** (thread &__t) noexcept

Static Public Member Functions

- static unsigned int **hardware_concurrency** () noexcept

4.972.1 Detailed Description

thread

Definition at line 60 of file thread.

4.972.2 Member Function Documentation

4.972.2.1 native_handle_type std::thread::native_handle ( ) [inline]

Precondition

thread is joinable

Definition at line 177 of file thread.

The documentation for this class was generated from the following file:

- thread

4.973 std::thread::id Class Reference

Public Member Functions

- **id** (native_handle_type __id)

Friends

- class **hash**< **thread::id** >
- bool **operator<** (thread::id __x, thread::id __y) noexcept
- template<class _CharT , class _Traits >
basic_ostream< _CharT, _Traits > & **operator<<** (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- bool **operator==** (thread::id __x, thread::id __y) noexcept
- class **thread**

4.973.1 Detailed Description

thread::id

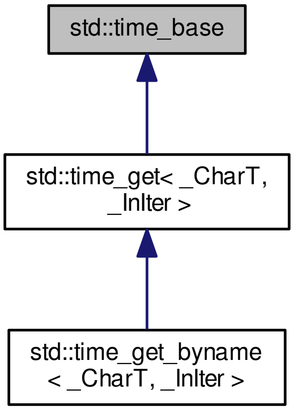
Definition at line 68 of file thread.

The documentation for this class was generated from the following file:

- thread

4.974 std::time_base Class Reference

Inheritance diagram for std::time_base:



Public Types

- enum **dateorder** {
  no_order, **dmy**, **mdy**, **ymd**,
  ydm }

4.974.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

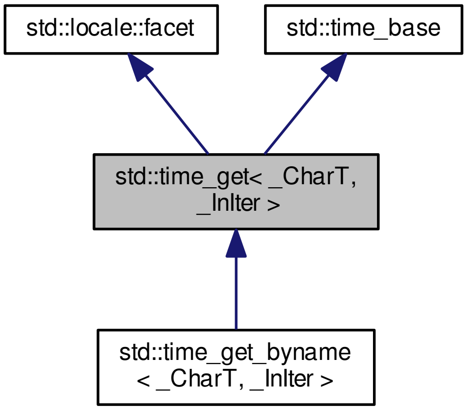
Definition at line 52 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

4.975 std::time_get< _CharT, _InIter > Class Template Reference

Inheritance diagram for std::time_get< _CharT, _InIter >:



Public Types

- typedef basic_string< _CharT > **__string_type**
- enum **dateorder** {
  no_order, **dmy**, **mdy**, **ymd**,
  ydm }

- typedef _CharT char_type
- typedef _InIter iter_type

Public Member Functions

- time_get (size_t __refs=0)
- dateorder date_order () const
- iter_type get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼time_get ()
- iter_type **_M_extract_name** (iter_type __beg, iter_type __end, int &__member, const _CharT ∗∗__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_num** (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm, const _CharT ∗__format) const
- iter_type **_M_extract_wday_or_month** (iter_type __beg, iter_type __end, int &__member, const _CharT ∗∗__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- virtual dateorder do_date_order () const
- virtual iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.975.1 Detailed Description

template<typename _CharT, typename _InIter>class std::time_get< _CharT, _InIter >

Primary class template time_get.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The time_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time_get facet.

Definition at line 368 of file locale_facets_nonio.h.

4.975.2 Member Typedef Documentation

4.975.2.1 template<typename _CharT , typename _InIter > typedef _CharT std::time_get< _CharT, _InIter >::char_type

Public typedefs.

Definition at line 374 of file locale_facets_nonio.h.

4.975.2.2 template<typename _CharT , typename _InIter > typedef _InIter std::time_get< _CharT, _InIter >::iter_type

Public typedefs.

Definition at line 375 of file locale_facets_nonio.h.

4.975.3 Constructor & Destructor Documentation

4.975.3.1 template<typename _CharT , typename _InIter > std::time_get< _CharT, _InIter >::time_get (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 390 of file locale_facets_nonio.h.

4.975.3.2 template<typename _CharT , typename _InIter > virtual std::time_get< _CharT, _InIter >::~time_get ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 546 of file locale_facets_nonio.h.

4.975.4 Member Function Documentation

4.975.4.1 template<typename _CharT , typename _InIter > dateorder std::time_get< _CharT, _InIter >::date_order ( ) const [inline]

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns timebase::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

Returns

A member of timebase::dateorder.

Definition at line 407 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_date_order().

4.975.4.2 template<typename _CharT , typename _InIter > time_base::dateorder std::time_get< _CharT, _InIter >::do_date_order ( ) const [protected], [virtual]

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of timebase::dateorder.

Definition at line 620 of file locale_facets_nonio.tcc.

Referenced by std::time_get< _CharT, _InIter >::date_order().

4.975.4.3 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_date (iter_type *__beg*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, tm * *__tm*) const [protected], [virtual]

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_date() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 1047 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

Referenced by std::time_get< _CharT, _InIter >::get_date().

4.975.4.4 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_monthname (iter_type *__beg*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, tm * *__tm*) const [protected], [virtual]

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_monthname() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1092 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

Referenced by std::time_get< _CharT, _InIter >::get_monthname().

4.975.4.5 template<typename _CharT , typename _InIter > _InIter **std::time_get< _CharT, _InIter >::do_get_time (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm * *__tm*) const [protected], [virtual]

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_time() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 1030 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

Referenced by std::time_get< _CharT, _InIter >::get_time().

4.975.4.6 template<typename _CharT , typename _InIter > _InIter **std::time_get< _CharT, _InIter >::do_get_weekday (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm * *__tm*) const [protected], [virtual]

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_weekday() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.

__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 1064 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

Referenced by std::time_get< _CharT, _InIter >::get_weekday().

4.975.4.7 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_year (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const `[protected]`, `[virtual]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_year() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 1120 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

Referenced by std::time_get< _CharT, _InIter >::get_year().

4.975.4.8 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_date (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const `[inline]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_date().

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 456 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_date().

4.975.4.9 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_monthname (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const [inline]

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_monthname().

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_monthname().

4.975.4.10 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_time (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const [inline]

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 431 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_time().

4.975.4.11 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_weekday (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const [inline]

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_weekday().

4.975.4.12 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_year (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const [inline]

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 539 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_year().

4.975.5 Member Data Documentation

4.975.5.1 template<typename _CharT , typename _InIter > locale::id std::time_get< _CharT, _InIter >::id [static]

Numpunct facet id.

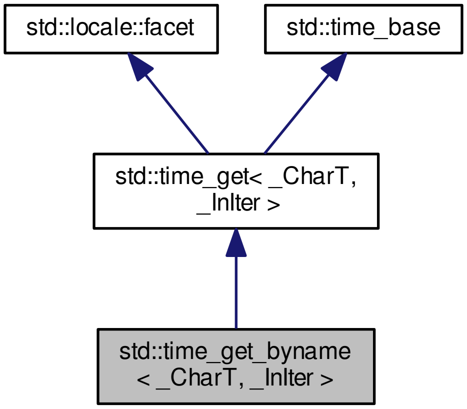
Definition at line 380 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

4.976 std::time_get_byname< _CharT, _InIter > Class Template Reference

Inheritance diagram for std::time_get_byname< _CharT, _InIter >:



Public Types

- typedef basic_string< _CharT > **__string_type**
- typedef _CharT **char_type**
- enum **dateorder** {
  no_order, **dmy**, **mdy**, **ymd**,
  ydm }
- typedef _InIter **iter_type**

Public Member Functions

- **time_get_byname** (const char ∗, size_t __refs=0)
- dateorder date_order () const
- iter_type get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- iter_type get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- iter_type get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- iter_type get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- iter_type **_M_extract_name** (iter_type __beg, iter_type __end, int &__member, const _CharT ∗∗__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_num** (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm, const _CharT ∗__format) const
- iter_type **_M_extract_wday_or_month** (iter_type __beg, iter_type __end, int &__member, const _CharT ∗∗__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- virtual dateorder do_date_order () const
- virtual iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm ∗__tm) const
- virtual iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm ∗__tm) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.976.1 Detailed Description

template<typename _CharT, typename _InIter>class std::time_get_byname< _CharT, _InIter >

class time_get_byname [22.2.5.2].

Definition at line 686 of file locale_facets_nonio.h.

4.976.2 Member Function Documentation

4.976.2.1 template<typename _CharT , typename _InIter > dateorder std::time_get< _CharT, _InIter >::date_order ( ) const [inline], [inherited]

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns timebase::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

Returns

A member of timebase::dateorder.

Definition at line 407 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_date_order().

4.976.2.2 template<typename _CharT , typename _InIter > time_base::dateorder std::time_get< _CharT, _InIter >::do_date_order ( ) const [protected], [virtual], [inherited]

Return preferred order of month, day, and year.

This function returns an enum from timebase::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of timebase::dateorder.

Definition at line 620 of file locale_facets_nonio.tcc.

Referenced by std::time_get< _CharT, _InIter >::date_order().

4.976.2.3 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_date (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [protected], [virtual], [inherited]

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_date() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 1047 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

Referenced by std::time_get< _CharT, _InIter >::get_date().

4.976.2.4 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_monthname (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, tm * __tm) const [protected], [virtual], [inherited]

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_monthname() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 1092 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

Referenced by std::time_get< _CharT, _InIter >::get_monthname().

4.976.2.5 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_time (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const [protected], [virtual], [inherited]

Parse input time string.

This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_time() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 1030 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::ios_base::eofbit.

Referenced by std::time_get< _CharT, _InIter >::get_time().

4.976.2.6 template<typename _CharT , typename _InIter > _InIter std::time_get< _CharT, _InIter >::do_get_weekday (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm ∗ *__tm*) const [protected], [virtual], [inherited]

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_weekday() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.

__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 1064 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

Referenced by std::time_get< _CharT, _InIter >::get_weekday().

4.976.2.7 template<typename _CharT , typename _InIter > _InIter **std::time_get**< _CharT, _InIter >::do_get_year (**iter_type** *__beg,* **iter_type** *__end,* **ios_base &** *__io,* **ios_base::iostate &** *__err,* tm * *__tm*) const `[protected]`, `[virtual]`, `[inherited]`

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See Also

get_year() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 1120 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, and std::ios_base::goodbit.

Referenced by std::time_get< _CharT, _InIter >::get_year().

4.976.2.8 template<typename _CharT , typename _InIter > iter_type **std::time_get**< _CharT, _InIter >::get_date (**iter_type** *__beg,* **iter_type** *__end,* **ios_base &** *__io,* **ios_base::iostate &** *__err,* tm * *__tm*) const `[inline]`, `[inherited]`

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_date().

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

Definition at line 456 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_date().

4.976.2.9 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_monthname (iter_type *__beg*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, tm * *__tm*) const [inline], [inherited]

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_monthname().

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

Definition at line 513 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_monthname().

4.976.2.10 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_time (iter_type *__beg*, iter_type *__end*, ios_base & *__io*, ios_base::iostate & *__err*, tm * *__tm*) const [inline], [inherited]

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

Definition at line 431 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_time().

4.976.2.11 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_weekday (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm * *__tm*) const [inline], [inherited]

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

Definition at line 484 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_weekday().

4.976.2.12 template<typename _CharT , typename _InIter > iter_type std::time_get< _CharT, _InIter >::get_year (iter_type *__beg,* iter_type *__end,* ios_base & *__io,* ios_base::iostate & *__err,* tm * *__tm*) const [inline], [inherited]

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

Definition at line 539 of file locale_facets_nonio.h.

References std::time_get< _CharT, _InIter >::do_get_year().

4.976.3 Member Data Documentation

4.976.3.1 template<typename _CharT , typename _InIter > locale::id std::time_get< _CharT, _InIter >::id `[static]`, `[inherited]`

Numpunct facet id.

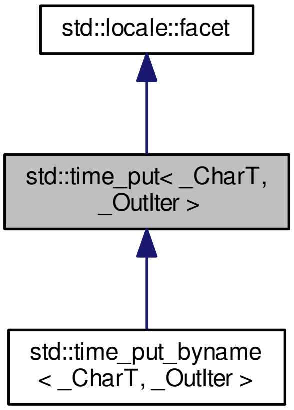
Definition at line 380 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

4.977 std::time_put< _CharT, _OutIter > Class Template Reference

Inheritance diagram for std::time_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type

Public Member Functions

- time_put (size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm ∗__tm, const _CharT ∗__beg, const _CharT ∗__end) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm ∗__tm, char __format, char __mod=0) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ∼time_put ()
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm ∗__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.977.1 Detailed Description

template<typename _CharT, typename _OutIter>class std::time_put< _CharT, _OutIter >

Primary class template time_put.

This facet encapsulates the code to format and output dates and times according to formats used by strftime().

The time_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time_put facet.

Definition at line 715 of file locale_facets_nonio.h.

4.977.2 Member Typedef Documentation

4.977.2.1 template<typename _CharT , typename _OutIter > typedef _CharT std::time_put< _CharT, _OutIter >::char_type

Public typedefs.

Definition at line 721 of file locale_facets_nonio.h.

4.977.2.2 template<typename _CharT , typename _OutIter > typedef _OutIter **std::time_put**< _CharT, _OutIter >**::iter_type**

Public typedefs.

Definition at line 722 of file locale_facets_nonio.h.

4.977.3 Constructor & Destructor Documentation

4.977.3.1 template<typename _CharT , typename _OutIter > **std::time_put**< _CharT, _OutIter >**::time_put** (size_t *__refs* = 0) [inline], [explicit]

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

Definition at line 736 of file locale_facets_nonio.h.

4.977.3.2 template<typename _CharT , typename _OutIter > virtual **std::time_put**< _CharT, _OutIter >**::~time_put** ( ) [inline], [protected], [virtual]

Destructor.

Definition at line 782 of file locale_facets_nonio.h.

4.977.4 Member Function Documentation

4.977.4.1 template<typename _CharT , typename _OutIter > _OutIter **std::time_put**< _CharT, _OutIter >::do_put (**iter_type** *__s*, **ios_base** & *__io*, **char_type** *__fill*, const tm * *__tm*, char *__format*, char *__mod*) const [protected], [virtual]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See Also

put() for more details.

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

Definition at line 1178 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::__ctype_abstract_base< _CharT >::widen().

Referenced by std::time_put< _CharT, _OutIter >::put().

4.977.4.2 template<typename _CharT , typename _OutIter > _OutIter std::time_put< _CharT, _OutIter >::put (iter_type *__s*, ios_base & *__io*, char_type *__fill*, const tm * *__tm*, const _CharT * *__beg*, const _CharT * *__end*) const

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__beg	Start of format string.
__end	End of format string.

Returns

Iterator after writing.

Definition at line 1143 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::__ctype_abstract_base< _CharT >::narrow().

4.977.4.3 template<typename _CharT , typename _OutIter > iter_type std::time_put< _CharT, _OutIter >::put (iter_type *__s*, ios_base & *__io*, char_type *__fill*, const tm * *__tm*, char *__format*, char *__mod* = `0`) const `[inline]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time_put::do_put().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

Definition at line 775 of file locale_facets_nonio.h.

References std::time_put< _CharT, _OutIter >::do_put().

4.977.5 Member Data Documentation

4.977.5.1 template<typename _CharT , typename _OutIter > **locale::id std::time_put< _CharT, _OutIter >::id** `[static]`

Numpunct facet id.

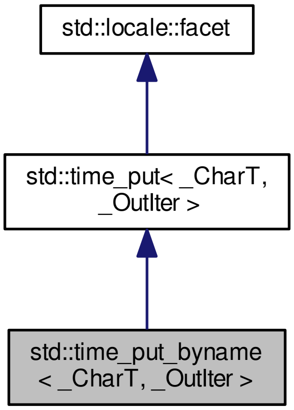
Definition at line 726 of file locale_facets_nonio.h.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

4.978 std::time_put_byname< _CharT, _OutIter > Class Template Reference

Inheritance diagram for std::time_put_byname< _CharT, _OutIter >:



Public Types

- typedef _CharT **char_type**
- typedef _OutIter **iter_type**

Public Member Functions

- **time_put_byname** (const char ∗, size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm ∗__tm, const _CharT ∗__beg, const _CharT ∗__end) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm ∗__tm, char __format, char __mod=0) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm ∗__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

4.978.1 Detailed Description

template<typename _CharT, typename _OutIter>class std::time_put_byname< _CharT, _OutIter >

class time_put_byname [22.2.5.4].

Definition at line 811 of file locale_facets_nonio.h.

4.978.2 Member Function Documentation

4.978.2.1 template<typename _CharT , typename _OutIter > _OutIter std::time_put< _CharT, _OutIter >::do_put (iter_type *__s,* ios_base & *__io,* char_type *__fill,* const tm ∗ *__tm,* char *__format,* char *__mod*) const `[protected]`, `[virtual]`, `[inherited]`

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See Also

put() for more details.

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

Definition at line 1178 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::__ctype_abstract_base< _CharT >::widen().

Referenced by std::time_put< _CharT, _OutIter >::put().

4.978.2.2 template<typename _CharT , typename _OutIter > _OutIter std::time_put< _CharT, _OutIter >::put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, const _CharT * __beg, const _CharT * __end) const [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__beg	Start of format string.
__end	End of format string.

Returns

Iterator after writing.

Definition at line 1143 of file locale_facets_nonio.tcc.

References std::ios_base::_M_getloc(), and std::__ctype_abstract_base< _CharT >::narrow().

4.978.2.3 template<typename _CharT , typename _OutIter > iter_type std::time_put< _CharT, _OutIter >::put (iter_type __s, ios_base & __io, char_type __fill, const tm * __tm, char __format, char __mod = 0) const [inline], [inherited]

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time_put::do_put().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

Definition at line 775 of file locale_facets_nonio.h.

References std::time_put< _CharT, _OutIter >::do_put().

4.978.3 Member Data Documentation

4.978.3.1 template<typename _CharT , typename _OutIter > locale::id std::time_put< _CharT, _OutIter >::id `[static]`, `[inherited]`

Numpunct facet id.

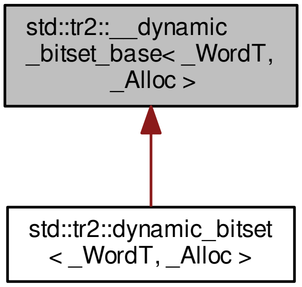
Definition at line 726 of file locale_facets_nonio.h.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

4.979 std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference

Inheritance diagram for std::tr2::__dynamic_bitset_base< _WordT, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _WordT **block_type**
- typedef size_t **size_type**

Public Member Functions

- **__dynamic_bitset_base** (const allocator_type &__alloc=allocator_type())
- **__dynamic_bitset_base** (__dynamic_bitset_base &&__b)

- **__dynamic_bitset_base** (size_type __nbits, unsigned long long __val=0ULL, const allocator_type &__alloc=allocator_type())
- size_t **_M_are_all_aux** () const
- void **_M_assign** (const __dynamic_bitset_base &__b)
- void **_M_clear** ()
- void **_M_do_and** (const __dynamic_bitset_base &__x)
- void **_M_do_append_block** (block_type __block, size_type __pos)
- size_t **_M_do_count** () const
- void **_M_do_dif** (const __dynamic_bitset_base &__x)
- size_type **_M_do_find_first** (size_t __not_found) const
- size_type **_M_do_find_next** (size_t __prev, size_t __not_found) const
- void **_M_do_flip** ()
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const __dynamic_bitset_base &__x)
- void **_M_do_reset** ()
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const __dynamic_bitset_base &__x)
- allocator_type **_M_get_allocator** () const
- block_type & **_M_getword** (size_type __pos)
- block_type **_M_getword** (size_type __pos) const
- block_type & **_M_hiword** ()
- block_type **_M_hiword** () const
- bool **_M_is_any** () const
- bool **_M_is_equal** (const __dynamic_bitset_base &__x) const
- bool **_M_is_less** (const __dynamic_bitset_base &__x) const
- bool **_M_is_proper_subset_of** (const __dynamic_bitset_base &__b) const
- bool **_M_is_subset_of** (const __dynamic_bitset_base &__b)
- void **_M_resize** (size_t __nbits, bool __value)
- size_type **_M_size** () const noexcept
- void **_M_swap** (__dynamic_bitset_base &__b)

Static Public Member Functions

- static block_type **_S_maskbit** (size_type __pos) noexcept
- static size_type **_S_whichbit** (size_type __pos) noexcept
- static size_type **_S_whichbyte** (size_type __pos) noexcept
- static size_type **_S_whichword** (size_type __pos) noexcept

Public Attributes

- std::vector< block_type, allocator_type > _M_w

Static Public Attributes

- static const size_type **_S_bits_per_block**
- static const size_type **npos**

4.979.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >

Base class, general case.

See documentation for dynamic_bitset.

Definition at line 80 of file dynamic_bitset.

4.979.2 Member Data Documentation

4.979.2.1 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w

0 is the least significant word.

Definition at line 93 of file dynamic_bitset.

The documentation for this struct was generated from the following file:

- dynamic_bitset

4.980 std::tr2::__reflection_typelist< _First, _Rest...> Struct Template Reference

Public Types

- typedef std::false_type **empty**

4.980.1 Detailed Description

template<typename _First, typename... _Rest>struct std::tr2::__reflection_typelist< _First, _Rest...>

Partial specialization.

Definition at line 67 of file tr2/type_traits.

The documentation for this struct was generated from the following file:

- tr2/type_traits

4.981 std::tr2::__reflection_typelist<> Struct Template Reference

Public Types

- typedef std::true_type **empty**

4.981.1 Detailed Description

template<>struct std::tr2::__reflection_typelist<>

Specialization for an empty typelist.

Definition at line 60 of file tr2/type_traits.

The documentation for this struct was generated from the following file:

- tr2/type_traits

4.982 std::tr2::bases< _Tp > Struct Template Reference

Public Types

- typedef __reflection_typelist
  < __bases(_Tp)...> **type**

4.982.1 Detailed Description

template<typename _Tp>struct std::tr2::bases< _Tp >

Sequence abstraction metafunctions for manipulating a typelist.

Enumerate all the base classes of a class. Form of a typelist.

Definition at line 88 of file tr2/type_traits.

The documentation for this struct was generated from the following file:

- tr2/type_traits

4.983 std::tr2::bool_set Class Reference

Public Member Functions

- constexpr bool_set ()
- constexpr bool_set (bool __t)
- bool **contains** (bool_set __b) const
- bool equals (bool_set __b) const
- bool is_emptyset () const
- bool is_indeterminate () const
- bool is_singleton () const
- operator bool () const

Static Public Member Functions

- static bool_set **emptyset** ()
- static bool_set **indeterminate** ()

Friends

- bool_set **operator!** (bool_set __b)
- bool_set **operator&** (bool_set __s, bool_set __t)
- template<typename CharT , typename Traits >
  std::basic_ostream< CharT,
  Traits > & **operator<<** (std::basic_ostream< CharT, Traits > &__out, bool_set __b)

- bool_set **operator==** (bool_set __s, bool_set __t)
- template<typename CharT , typename Traits >
  std::basic_istream< CharT,
  Traits > & **operator>>** (std::basic_istream< CharT, Traits > &__in, bool_set &__b)
- bool_set **operator^** (bool_set __s, bool_set __t)
- bool_set **operator|** (bool_set __s, bool_set __t)

4.983.1 Detailed Description

bool_set

See N2136, Bool_set: multi-valued logic by Hervnnimann, Guillaume Melquiond, Sylvain Pion.

The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

Definition at line 54 of file bool_set.

4.983.2 Constructor & Destructor Documentation

4.983.2.1 constexpr std::tr2::bool_set::bool_set ( ) `[inline]`

Default constructor.

Definition at line 59 of file bool_set.

4.983.2.2 constexpr std::tr2::bool_set::bool_set (bool *__t*) `[inline]`

Constructor from bool.

Definition at line 62 of file bool_set.

4.983.3 Member Function Documentation

4.983.3.1 bool std::tr2::bool_set::equals (bool_set *__b*) const `[inline]`

Return true if states are equal.

Definition at line 69 of file bool_set.

4.983.3.2 bool std::tr2::bool_set::is_emptyset ( ) const `[inline]`

Return true if this is empty.

Definition at line 73 of file bool_set.

4.983.3.3 bool std::tr2::bool_set::is_indeterminate ( ) const `[inline]`

Return true if this is indeterminate.

Definition at line 77 of file bool_set.

4.983.3.4 bool std::tr2::bool_set::is_singleton ( ) const `[inline]`

Return true if this is false or true (normal boolean).

Definition at line 81 of file bool_set.

Referenced by operator bool().

4.983.3.5 std::tr2::bool_set::operator bool ( ) const [inline]

Conversion to bool.

Definition at line 86 of file bool_set.

References is_singleton().

The documentation for this class was generated from the following files:

- bool_set
- bool_set.tcc

4.984 std::tr2::direct_bases< _Tp > Struct Template Reference

Public Types

- typedef __reflection_typelist
  < __direct_bases(_Tp)...> **type**

4.984.1 Detailed Description

template<typename _Tp>struct std::tr2::direct_bases< _Tp >

Enumerate all the direct base classes of a class. Form of a typelist.

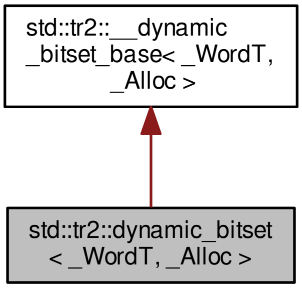
Definition at line 95 of file tr2/type_traits.

The documentation for this struct was generated from the following file:

- tr2/type_traits

4.985 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference

Inheritance diagram for std::tr2::dynamic_bitset< _WordT, _Alloc >:



Classes

- class reference

Public Types

- typedef __dynamic_bitset_base
  < _WordT, _Alloc > **_Base**
- typedef _Alloc **allocator_type**
- typedef _WordT **block_type**
- typedef bool **const_reference**
- typedef size_t **size_type**

Public Member Functions

- dynamic_bitset (const allocator_type &__alloc=allocator_type())
- dynamic_bitset (size_type __nbits, unsigned long long __val=0ULL, const allocator_type &__alloc=allocator_type())
- **dynamic_bitset** (initializer_list< block_type > __il, const allocator_type &__alloc=allocator_type())
- template<typename _CharT , typename _Traits , typename _Alloc1 >
  dynamic_bitset (const std::basic_string< _CharT, _Traits, _Alloc1 > &__str, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos=0, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n=std::basic_string< _CharT, _Traits, _Alloc1 >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'), const allocator_type &__alloc=allocator_type())
- dynamic_bitset (const char ∗__str, const allocator_type &__alloc=allocator_type())
- dynamic_bitset (const dynamic_bitset &__b)
- dynamic_bitset (dynamic_bitset &&__b)
- template<typename _CharT , typename _Traits >
  void **_M_copy_from_ptr** (const _CharT ∗, size_t, size_t, size_t, _CharT, _CharT)
- template<typename _CharT , typename _Traits , typename _Alloc1 >
  void **_M_copy_from_string** (const std::basic_string< _CharT, _Traits, _Alloc1 > &__str, size_t __pos, size_t __n, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<typename _CharT , typename _Traits , typename _Alloc1 >
  void **_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc1 > &__str, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- bool all () const
- bool any () const
- void append (block_type __block)
- void **append** (initializer_list< block_type > __il)
- template<typename _BlockInputIterator >
  void append (_BlockInputIterator __first, _BlockInputIterator __last)
- void clear ()
- size_type count () const noexcept
- bool empty () const noexcept
- size_type find_first () const
- size_type find_next (size_t __prev) const
- dynamic_bitset< _WordT, _Alloc > & flip ()
- dynamic_bitset< _WordT, _Alloc > & flip (size_type __pos)
- allocator_type get_allocator () const
- bool **is_proper_subset_of** (const dynamic_bitset &__b) const
- bool **is_subset_of** (const dynamic_bitset &__b) const

- constexpr size_type max_size () noexcept
- bool none () const
- size_type num_blocks () const noexcept
- dynamic_bitset & operator= (const dynamic_bitset &__b)
- dynamic_bitset & operator= (dynamic_bitset &&__b)
- dynamic_bitset< _WordT, _Alloc > operator∼ () const
- void push_back (bool __bit)
- dynamic_bitset< _WordT, _Alloc > & reset ()
- dynamic_bitset< _WordT, _Alloc > & reset (size_type __pos)
- void resize (size_type __nbits, bool __value=false)
- dynamic_bitset< _WordT, _Alloc > & set ()
- dynamic_bitset< _WordT, _Alloc > & set (size_type __pos, bool __val=true)
- size_type size () const noexcept
- void swap (dynamic_bitset &__b)
- bool test (size_type __pos) const
- template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 = std::allocator<_CharT>>
  std::basic_string< _CharT,
  _Traits, _Alloc1 > to_string (_CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- unsigned long long to_ullong () const
- unsigned long to_ulong () const

- dynamic_bitset< _WordT, _Alloc > & operator&= (const dynamic_bitset< _WordT, _Alloc > &__rhs)
- dynamic_bitset< _WordT, _Alloc > & operator&= (dynamic_bitset< _WordT, _Alloc > &&__rhs)
- dynamic_bitset< _WordT, _Alloc > & operator|= (const dynamic_bitset< _WordT, _Alloc > &__rhs)
- dynamic_bitset< _WordT, _Alloc > & operator^= (const dynamic_bitset< _WordT, _Alloc > &__rhs)
- dynamic_bitset< _WordT, _Alloc > & operator-= (const dynamic_bitset< _WordT, _Alloc > &__rhs)

- dynamic_bitset< _WordT, _Alloc > & operator<<= (size_type __pos)
- dynamic_bitset< _WordT, _Alloc > & operator>>= (size_type __pos)

- reference operator[] (size_type __pos)
- const_reference operator[] (size_type __pos) const

- dynamic_bitset< _WordT, _Alloc > operator<< (size_type __pos) const
- dynamic_bitset< _WordT, _Alloc > operator>> (size_type __pos) const

Static Public Attributes

- static const size_type **bits_per_block**
- static const size_type **npos**

Private Member Functions

- size_t **_M_are_all_aux** () const
- void **_M_assign** (const __dynamic_bitset_base &__b)
- void **_M_clear** ()
- void **_M_do_and** (const __dynamic_bitset_base &__x)
- void **_M_do_append_block** (block_type __block, size_type __pos)
- size_t **_M_do_count** () const
- void **_M_do_dif** (const __dynamic_bitset_base &__x)

- size_type **_M_do_find_first** (size_t __not_found) const
- size_type **_M_do_find_next** (size_t __prev, size_t __not_found) const
- void **_M_do_flip** ()
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const __dynamic_bitset_base &__x)
- void **_M_do_reset** ()
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** ()
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const __dynamic_bitset_base &__x)
- allocator_type **_M_get_allocator** () const
- block_type & **_M_getword** (size_type __pos)
- block_type **_M_getword** (size_type __pos) const
- block_type & **_M_hiword** ()
- block_type **_M_hiword** () const
- bool **_M_is_any** () const
- bool **_M_is_equal** (const __dynamic_bitset_base &__x) const
- bool **_M_is_less** (const __dynamic_bitset_base &__x) const
- bool **_M_is_proper_subset_of** (const __dynamic_bitset_base &__b) const
- bool **_M_is_subset_of** (const __dynamic_bitset_base &__b)
- void **_M_resize** (size_t __nbits, bool __value)
- size_type **_M_size** () const noexcept
- void **_M_swap** (__dynamic_bitset_base &__b)

Static Private Member Functions

- static block_type **_S_maskbit** (size_type __pos) noexcept
- static size_type **_S_whichbit** (size_type __pos) noexcept
- static size_type **_S_whichbyte** (size_type __pos) noexcept
- static size_type **_S_whichword** (size_type __pos) noexcept

Private Attributes

- std::vector< block_type,
  allocator_type > _M_w

Static Private Attributes

- static const size_type **_S_bits_per_block**

Friends

- class **reference**

4.985.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>class std::tr2::dynamic_bitset< _WordT, _Alloc >

The dynamic_bitset class represents a sequence of bits.

(Note that dynamic_bitset does *not* meet the formal requirements of a `container`. Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024∗1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then (Nb+(B-1))/B words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of dynamic_bitset as "a simple array of bits," be aware that your mental picture is reversed: a dynamic_bitset behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and the bit at index Nb-1 in the "most significant / left-hand" position. Thus, unlike other containers, a dynamic_bitset's index "counts from right to left," to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long         a = 'a';
    dynamic_bitset   b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string  str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch33s02.html` for a description of extensions.

Most of the actual code isn't contained in dynamic_bitset<> itself, but in the base class __dynamic_bitset_base. The base class works with whole words, not with individual bits. This allows us to specialize __dynamic_bitset_base for the important special case where the dynamic_bitset is only a single word.

Extra confusion can result due to the fact that the storage for __dynamic_bitset_base *is* a vector, and is indexed as such. This is carefully encapsulated.

Definition at line 569 of file dynamic_bitset.

4.985.2 Constructor & Destructor Documentation

4.985.2.1 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const allocator_type & *__alloc* = `allocator_type()`) `[inline]`, `[explicit]`

All bits set to zero.

Definition at line 721 of file dynamic_bitset.

4.985.2.2 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (size_type __nbits, unsigned long long __val = `0ULL`, const allocator_type & __alloc = `allocator_type()`) `[inline]`, `[explicit]`

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 727 of file dynamic_bitset.

4.985.2.3 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> template<typename _CharT , typename _Traits , typename _Alloc1 > std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const std::basic_string< _CharT, _Traits, _Alloc1 > & __str, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = `0`, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<`_CharT, _Traits, _Alloc1`>`::npos`, _CharT __zero = `_CharT('0')`, _CharT __one = `_CharT('1')`, const allocator_type & __alloc = `allocator_type()`) `[inline]`, `[explicit]`

Use a subset of a string.

Parameters

__str	A string of '0' and '1' characters.
__pos	Index of the first character in `__str` to use.
__n	The number of characters to copy.

Exceptions

std::out_of_range	If `__pos` is bigger the size of `__str`.
std::invalid_argument	If a character appears in the string which is neither '0' nor '1'.

Definition at line 749 of file dynamic_bitset.

References std::tr2::dynamic_bitset< _WordT, _Alloc >::resize(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

4.985.2.4 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const char * __str, const allocator_type & __alloc = `allocator_type()`) `[inline]`, `[explicit]`

Construct from a string.

Parameters

__str	A string of '0' and '1' characters.

Exceptions

std::invalid_argument	If a character appears in the string which is neither '0' nor '1'.

Definition at line 777 of file dynamic_bitset.

References std::tr2::dynamic_bitset< _WordT, _Alloc >::resize().

4.985.2.5 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (const dynamic_bitset< _WordT, _Alloc > & __b) `[inline]`

Copy constructor.

Definition at line 793 of file dynamic_bitset.

4.985.2.6 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (dynamic_bitset< _WordT, _Alloc > && *__b*) [inline]

Move constructor.

Definition at line 800 of file dynamic_bitset.

4.985.3 Member Function Documentation

4.985.3.1 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all ( ) const [inline]

Tests whether all the bits are on.

Returns

True if all the bits are set.

Definition at line 1187 of file dynamic_bitset.

4.985.3.2 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any ( ) const [inline]

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

Definition at line 1195 of file dynamic_bitset.

4.985.3.3 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (block_type *__block*) [inline]

Append a block.

Definition at line 881 of file dynamic_bitset.

Referenced by std::tr2::dynamic_bitset< _WordT, _Alloc >::append().

4.985.3.4 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> template<typename _BlockInputIterator > void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (_BlockInputIterator *__first,* _BlockInputIterator *__last*) [inline]

Append an iterator range of blocks.

Definition at line 899 of file dynamic_bitset.

References std::tr2::dynamic_bitset< _WordT, _Alloc >::append().

4.985.3.5 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void std::tr2::dynamic_bitset< _WordT, _Alloc >::clear ( ) [inline]

Clear the bitset.

Definition at line 859 of file dynamic_bitset.

4.985.3.6 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::count ( ) const [inline], [noexcept]

Returns the number of bits which are set.

Definition at line 1143 of file dynamic_bitset.

4.985.3.7 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool std::tr2::dynamic_bitset< _WordT, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the dynamic_bitset is empty.

Definition at line 1158 of file dynamic_bitset.

4.985.3.8 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_first ( ) const [inline]

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See Also

find_next

Definition at line 1223 of file dynamic_bitset.

4.985.3.9 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_next (size_t *__prev*) const [inline]

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

__prev	Where to start searching.

See Also

find_first

Definition at line 1233 of file dynamic_bitset.

4.985.3.10 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::flip ( ) [inline]

Toggles every bit to its opposite value.

Definition at line 1038 of file dynamic_bitset.

Referenced by std::tr2::dynamic_bitset< _WordT, _Alloc >::operator∼().

4.985.3.11 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::flip (size_type __pos) [inline]

Toggles a given bit to its opposite value.

Parameters

__pos	The index of the bit.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

Definition at line 1051 of file dynamic_bitset.

4.985.3.12 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> allocator_type std::tr2::dynamic_bitset< _WordT, _Alloc >::get_allocator ( ) const [inline]

Return the allocator for the bitset.

Definition at line 841 of file dynamic_bitset.

4.985.3.13 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> constexpr size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size ( ) [inline], [noexcept]

Returns the maximum size of a dynamic_bitset object having the same type as ∗this. The real answer is max() ∗ bits_per_block but is likely to overflow.

Definition at line 1165 of file dynamic_bitset.

References std::numeric_limits< _Tp >::max().

4.985.3.14 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool std::tr2::dynamic_bitset< _WordT, _Alloc >::none ( ) const [inline]

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

Definition at line 1203 of file dynamic_bitset.

4.985.3.15 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::num_blocks ( ) const [inline], [noexcept]

Returns the total number of blocks.

Definition at line 1153 of file dynamic_bitset.

4.985.3.16 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

Definition at line 914 of file dynamic_bitset.

4.985.3.17 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (dynamic_bitset< _WordT, _Alloc > && *__rhs*) [inline]

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

Definition at line 921 of file dynamic_bitset.

4.985.3.18 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator-= (const dynamic_bitset< _WordT, _Alloc > & *__rhs*) [inline]

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

Definition at line 942 of file dynamic_bitset.

4.985.3.19 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc> std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<< (size_type *__pos*) const [inline]

Self-explanatory.

Definition at line 1209 of file dynamic_bitset.

4.985.3.20 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (size_type *__pos*) [inline]

Operations on dynamic_bitsets.

Parameters

__pos	The number of places to shift.

These should be self-explanatory.

Definition at line 957 of file dynamic_bitset.

4.985.3.21 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (const dynamic_bitset< _WordT, _Alloc > & __b) [inline]

Assignment.

Definition at line 818 of file dynamic_bitset.

4.985.3.22 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (dynamic_bitset< _WordT, _Alloc > && __b) [inline]

Move assignment.

Definition at line 831 of file dynamic_bitset.

References std::tr2::dynamic_bitset< _WordT, _Alloc >::swap().

4.985.3.23 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc> std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (size_type __pos) const [inline]

Self-explanatory.

Definition at line 1213 of file dynamic_bitset.

4.985.3.24 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (size_type __pos) [inline]

Operations on dynamic_bitsets.

Parameters

__pos	The number of places to shift.

These should be self-explanatory.

Definition at line 970 of file dynamic_bitset.

4.985.3.25 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (size_type __pos) [inline]

Array-indexing support.

Parameters

__pos	Index into the dynamic_bitset.

Returns

A bool for a 'const dynamic_bitset'. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 1073 of file dynamic_bitset.

4.985.3.26 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> const_reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (size_type __pos) const [inline]

Array-indexing support.

Parameters

__pos	Index into the dynamic_bitset.

Returns

A bool for a 'const dynamic_bitset'. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 1077 of file dynamic_bitset.

4.985.3.27 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator^= (const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

Definition at line 935 of file dynamic_bitset.

4.985.3.28 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator|= (const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

Definition at line 928 of file dynamic_bitset.

4.985.3.29 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc> std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ ( ) const [inline]

See the no-argument flip().

Definition at line 1060 of file dynamic_bitset.

References std::tr2::dynamic_bitset< _WordT, _Alloc >::flip().

4.985.3.30 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (bool __bit) [inline]

Push a bit onto the high end of the bitset.

Definition at line 869 of file dynamic_bitset.

References std::tr2::dynamic_bitset< _WordT, _Alloc >::size().

4.985.3.31 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset ( ) [inline]

Sets every bit to false.

Definition at line 1013 of file dynamic_bitset.

4.985.3.32 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (size_type __pos) [inline]

Sets a given bit to false.

Parameters

__pos	The index of the bit.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

Same as writing `set(__pos, false)`.

Definition at line 1027 of file dynamic_bitset.

4.985.3.33 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void std::tr2::dynamic_bitset< _WordT, _Alloc >::resize (size_type __nbits, bool __value = false) [inline]

Resize the bitset.

Definition at line 848 of file dynamic_bitset.

Referenced by std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset(), and std::tr2::operator>>().

4.985.3.34 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::set ( ) [inline]

Sets every bit to true.

Definition at line 988 of file dynamic_bitset.

4.985.3.35 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> dynamic_bitset<_WordT, _Alloc>& std::tr2::dynamic_bitset< _WordT, _Alloc >::set (size_type __pos, bool __val = true) [inline]

Sets a given bit to a particular value.

Parameters

__pos	The index of the bit.
__val	Either true or false, defaults to true.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

Definition at line 1002 of file dynamic_bitset.

4.985.3.36 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::size ( ) const [inline], [noexcept]

Returns the total number of bits.

Definition at line 1148 of file dynamic_bitset.

Referenced by std::tr2::operator>>(), and std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back().

4.985.3.37 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> void std::tr2::dynamic_bitset< _WordT, _Alloc >::swap (dynamic_bitset< _WordT, _Alloc > & __b) [inline]

Swap with another bitset.

Definition at line 808 of file dynamic_bitset.

Referenced by std::tr2::dynamic_bitset< _WordT, _Alloc >::operator=().

4.985.3.38 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> bool std::tr2::dynamic_bitset< _WordT, _Alloc >::test (size_type __pos) const [inline]

Tests the value of a bit.

Parameters

__pos	The index of a bit.

Returns

The value at *__pos*.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

Definition at line 1175 of file dynamic_bitset.

4.985.3.39 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 = std::allocator<_CharT>> std::basic_string<_CharT, _Traits, _Alloc1> std::tr2::dynamic_bitset< _WordT, _Alloc >::to_string (_CharT __zero = _CharT('0'), _CharT __one = _CharT('1')) const [inline]

Returns a character interpretation of the dynamic_bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1113 of file dynamic_bitset.

4.985.3.40 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> unsigned long long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ullong ( ) const [inline]

Returns a numerical interpretation of the dynamic_bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

Definition at line 1098 of file dynamic_bitset.

4.985.3.41 template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>> unsigned long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ulong ( ) const [inline]

Returns a numerical interpretation of the dynamic_bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

Definition at line 1088 of file dynamic_bitset.

The documentation for this class was generated from the following file:

- dynamic_bitset

4.986 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference

Public Member Functions

- **reference** (dynamic_bitset &__b, size_type __pos)
- reference & **flip** ()
- **operator bool** () const
- reference & **operator=** (bool __x)
- reference & **operator=** (const reference &__j)
- bool **operator∼** () const

Friends

- class **dynamic_bitset**

4.986.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)

Definition at line 654 of file dynamic_bitset.

The documentation for this class was generated from the following file:

- dynamic_bitset

4.987 std::try_to_lock_t Struct Reference

4.987.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

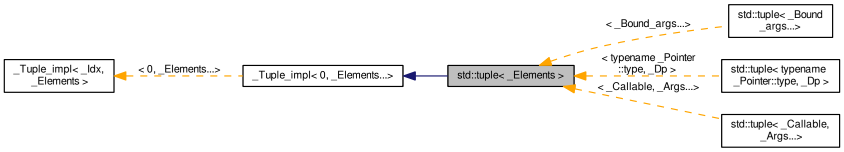
Definition at line 392 of file mutex.

The documentation for this struct was generated from the following file:

- mutex

4.988 std::tuple< _Elements > Class Template Reference

Inheritance diagram for std::tuple< _Elements >:



Public Member Functions

- constexpr **tuple** (const _Elements &... __elements)
- template<typename... _UElements, typename = typename enable_if<__and_<is_convertible<_UElements, _Elements>...>::value>::type>
  constexpr **tuple** (_UElements &&... __elements)
- constexpr **tuple** (const tuple &)=default
- constexpr **tuple** (tuple &&)=default
- template<typename... _UElements, typename = typename enable_if<__and_<is_convertible<const _UElements&, _Elements>...>::value>::type>
  constexpr **tuple** (const tuple< _UElements...> &__in)

- template<typename... _UElements, typename = typename enable_if<__and_<is_convertible<_UElements, _Elements>...>::value>::type>
  constexpr **tuple** (tuple< _UElements...> &&__in)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a, const _Elements &...__elements)
- template<typename _Alloc , typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
  tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&...__elements)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc , typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
  tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements...> &__in)
- template<typename _Alloc , typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
  tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements...> &&__in)
- tuple & **operator=** (const tuple &__in)
- tuple & **operator=** (tuple &&__in) noexcept(is_nothrow_move_assignable< _Inherited >
- template<typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
  tuple & **operator=** (const tuple< _UElements...> &__in)
- template<typename... _UElements, typename = typename enable_if<sizeof...(_UElements) == sizeof...(_Elements)>::type>
  tuple & **operator=** (tuple< _UElements...> &&__in)
- void **swap** (tuple &__in) noexcept(noexcept(__in._M_swap(__in)))

4.988.1 Detailed Description

template<typename... _Elements>class std::tuple< _Elements >

Primary class template, tuple.

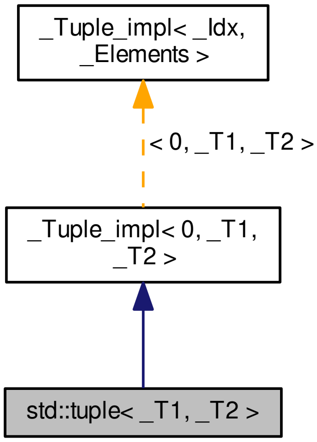
Definition at line 388 of file tuple.

The documentation for this class was generated from the following file:

- tuple

4.989 std::tuple< _T1, _T2 > Class Template Reference

Inheritance diagram for std::tuple< _T1, _T2 >:



Public Member Functions

- constexpr **tuple** (const _T1 &__a1, const _T2 &__a2)
- template<typename _U1 , typename _U2 , typename = typename enable_if<__and_<is_convertible<_U1, _T1>, is_convertible<_U2, _T2>>::value>::type>
  constexpr **tuple** (_U1 &&__a1, _U2 &&__a2)
- constexpr **tuple** (const tuple &)=default
- constexpr **tuple** (tuple &&)=default
- template<typename _U1 , typename _U2 , typename = typename enable_if<__and_<is_convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type>
  constexpr **tuple** (const tuple< _U1, _U2 > &__in)
- template<typename _U1 , typename _U2 , typename = typename enable_if<__and_<is_convertible<_U1, _T1>, is_convertible<_U2, _T2>>::value>::type>
  constexpr **tuple** (tuple< _U1, _U2 > &&__in)
- template<typename _U1 , typename _U2 , typename = typename enable_if<__and_<is_convertible<const _U1&, _T1>, is_convertible<const _U2&, _T2>>::value>::type>
  constexpr **tuple** (const pair< _U1, _U2 > &__in)
- template<typename _U1 , typename _U2 , typename = typename enable_if<__and_<is_convertible<_U1, _T1>, is_convertible<_U2, _T2>>::value>::type>
  constexpr **tuple** (pair< _U1, _U2 > &&__in)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)
- template<typename _Alloc , typename _U1 , typename _U2 >
  tuple (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)

- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)
- template<typename _Alloc >
  tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc , typename _U1 , typename _U2 >
  tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)
- template<typename _Alloc , typename _U1 , typename _U2 >
  tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)
- template<typename _Alloc , typename _U1 , typename _U2 >
  tuple (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)
- template<typename _Alloc , typename _U1 , typename _U2 >
  tuple (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)
- tuple & **operator=** (const tuple &__in)
- tuple & **operator=** (tuple &&__in) noexcept(is_nothrow_move_assignable< _Inherited >
- template<typename _U1 , typename _U2 >
  tuple & **operator=** (const tuple< _U1, _U2 > &__in)
- template<typename _U1 , typename _U2 >
  tuple & **operator=** (tuple< _U1, _U2 > &&__in)
- template<typename _U1 , typename _U2 >
  tuple & **operator=** (const pair< _U1, _U2 > &__in)
- template<typename _U1 , typename _U2 >
  tuple & **operator=** (pair< _U1, _U2 > &&__in)
- void **swap** (tuple &__in) noexcept(noexcept(__in._M_swap(__in)))

4.989.1 Detailed Description

template<typename _T1, typename _T2>class std::tuple< _T1, _T2 >

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.

Definition at line 521 of file tuple.

The documentation for this class was generated from the following file:

- tuple

4.990 std::tuple_element< 0, tuple< _Head, _Tail...> > Struct Template Reference

Public Types

- typedef _Head **type**

4.990.1 Detailed Description

template<typename _Head, typename... _Tail>struct std::tuple_element< 0, tuple< _Head, _Tail...> >

Basis case for tuple_element: The first element is the one we're seeking.

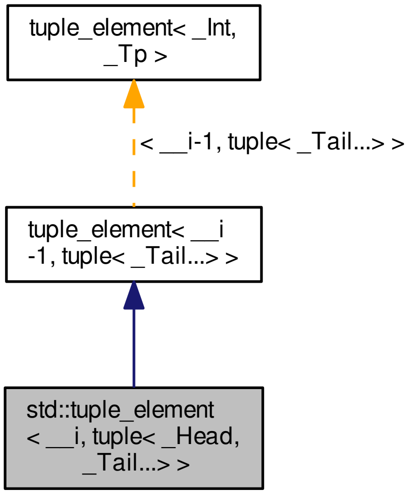
Definition at line 687 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.991 std::tuple_element< __i, tuple< _Head, _Tail...> > Struct Template Reference

Inheritance diagram for std::tuple_element< __i, tuple< _Head, _Tail...> >:



4.991.1 Detailed Description

template<std::size_t __i, typename _Head, typename... _Tail>struct std::tuple_element< __i, tuple< _Head, _Tail...> >

Recursive case for tuple_element: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

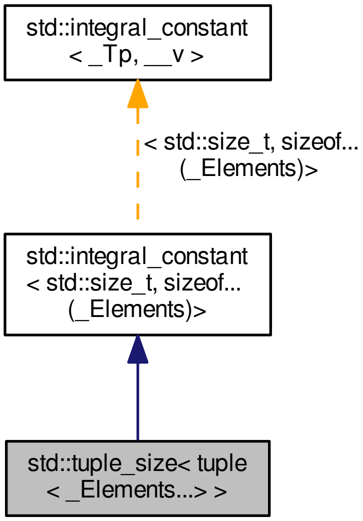
Definition at line 680 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.992 std::tuple_size< tuple< _Elements...> > Struct Template Reference

Inheritance diagram for std::tuple_size< tuple< _Elements...> >:



Public Types

- typedef integral_constant
  < std::size_t, __v > **type**
- typedef std::size_t **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr std::size_t **value**

4.992.1 Detailed Description

template<typename... _Elements>struct std::tuple_size< tuple< _Elements...> >

class tuple_size

Definition at line 737 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.993 std::type_index Struct Reference

Public Member Functions

- **type_index** (const type_info &__rhs) noexcept
- size_t **hash_code** () const noexcept
- const char ∗ **name** () const
- bool **operator!=** (const type_index &__rhs) const noexcept
- bool **operator<** (const type_index &__rhs) const noexcept
- bool **operator<=** (const type_index &__rhs) const noexcept
- bool **operator==** (const type_index &__rhs) const noexcept
- bool **operator>** (const type_index &__rhs) const noexcept
- bool **operator>=** (const type_index &__rhs) const noexcept

4.993.1 Detailed Description

Class type_index

The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Definition at line 52 of file typeindex.

The documentation for this struct was generated from the following file:

- typeindex

4.994 std::type_info Class Reference

Inherited by __cxxabiv1::__array_type_info, __cxxabiv1::__class_type_info, __cxxabiv1::__enum_type_info, __cxxabiv1::__function_type_info, __cxxabiv1::__fundamental_type_info, and __cxxabiv1::__pbase_type_info.

Public Member Functions

- virtual ∼type_info ()
- virtual bool **__do_catch** (const type_info ∗__thr_type, void ∗∗__thr_obj, unsigned __outer) const
- virtual bool **__do_upcast** (const __cxxabiv1::__class_type_info ∗__target, void ∗∗__obj_ptr) const
- virtual bool **__is_function_p** () const
- virtual bool **__is_pointer_p** () const
- bool **before** (const type_info &__arg) const noexcept
- size_t **hash_code** () const noexcept
- const char ∗ name () const noexcept
- bool **operator!=** (const type_info &__arg) const noexcept
- bool **operator==** (const type_info &__arg) const noexcept

Protected Member Functions

- **type_info** (const char ∗__n)

Protected Attributes

- const char ∗ **__name**

4.994.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

Definition at line 88 of file typeinfo.

4.994.2 Constructor & Destructor Documentation

4.994.2.1 virtual std::type_info::∼type_info ( ) `[virtual]`

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated type_info structures in the new-abi.

4.994.3 Member Function Documentation

4.994.3.1 const char∗ std::type_info::name ( ) const `[inline]`, `[noexcept]`

Returns an *implementation-defined* byte string; this is not portable between compilers!

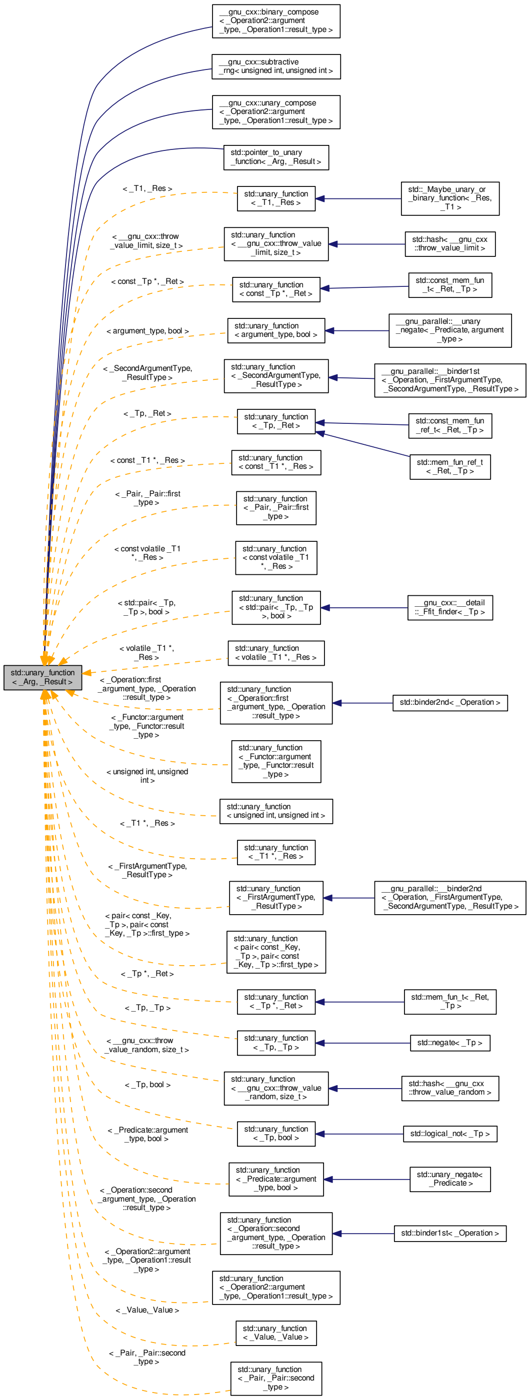
Definition at line 99 of file typeinfo.

The documentation for this class was generated from the following file:

- typeinfo

4.995 std::unary_function< _Arg, _Result > Struct Template Reference

Inheritance diagram for std::unary_function< _Arg, _Result >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

4.995.1 Detailed Description

template<typename _Arg, typename _Result>struct std::unary_function< _Arg, _Result >

This is one of the functor base classes.

Definition at line 101 of file stl_function.h.

4.995.2 Member Typedef Documentation

4.995.2.1 template<typename _Arg, typename _Result> typedef _Arg std::unary_function< _Arg, _Result >::argument_type

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.995.2.2 template<typename _Arg, typename _Result> typedef _Result std::unary_function< _Arg, _Result >::result_type

`result_type` is the return type

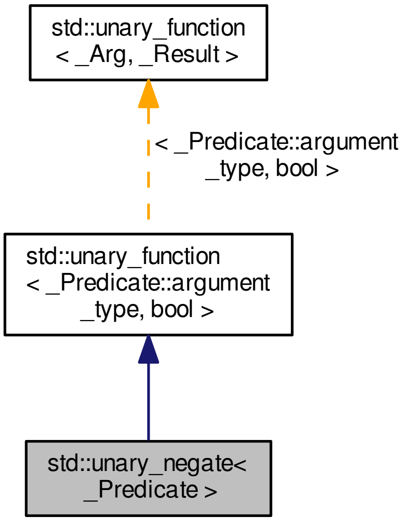
Definition at line 107 of file stl_function.h.

The documentation for this struct was generated from the following file:

- stl_function.h

4.996 std::unary_negate< _Predicate > Class Template Reference

Inheritance diagram for std::unary_negate< _Predicate >:



Public Types

- typedef _Predicate::argument_type argument_type
- typedef bool result_type

Public Member Functions

- **unary_negate** (const _Predicate &__x)
- bool **operator()** (const typename _Predicate::argument_type &__x) const

Protected Attributes

- _Predicate **_M_pred**

4.996.1 Detailed Description

template<typename _Predicate>class std::unary_negate< _Predicate >

One of the negation functors.

Definition at line 351 of file stl_function.h.

4.996.2 Member Typedef Documentation

4.996.2.1 typedef _Predicate::argument_type **std::unary_function**< _Predicate::argument_type , bool >**::argument_type** `[inherited]`

`argument_type` is the type of the argument

Definition at line 104 of file stl_function.h.

4.996.2.2 typedef bool **std::unary_function**< _Predicate::argument_type , bool >**::result_type** `[inherited]`

`result_type` is the return type

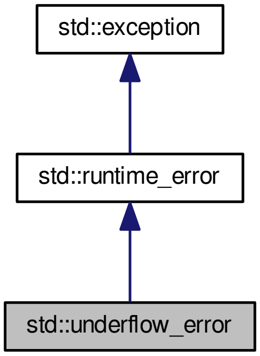
Definition at line 107 of file stl_function.h.

The documentation for this class was generated from the following file:

- stl_function.h

4.997 std::underflow_error Class Reference

Inheritance diagram for std::underflow_error:



Public Member Functions

- **underflow_error** (const string &__arg)
- virtual const char ∗ what () const noexcept

4.997.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 146 of file stdexcept.

4.997.2 Member Function Documentation

4.997.2.1 virtual const char∗ std::runtime_error::what ( ) const [virtual], [noexcept], [inherited]

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.

The documentation for this class was generated from the following file:

- stdexcept

4.998 std::underlying_type< _Tp > Struct Template Reference

Public Member Functions

- typedef **__underlying_type** (_Tp) type

4.998.1 Detailed Description

template<typename _Tp>struct std::underlying_type< _Tp >

The underlying type of an enum.

Definition at line 1855 of file type_traits.

The documentation for this struct was generated from the following file:

- type_traits

4.999 std::uniform_int_distribution< _IntType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- uniform_int_distribution (_IntType __a=0, _IntType __b=std::numeric_limits< _IntType >::max())
- **uniform_int_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- result_type **a** () const
- result_type **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const uniform_int_distribution &__d1, const uniform_int_distribution &__d2)

4.999.1 Detailed Description

template<typename _IntType = int>class std::uniform_int_distribution< _IntType >

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Definition at line 1666 of file random.h.

4.999.2 Member Typedef Documentation

4.999.2.1 template<typename _IntType = int> typedef _IntType **std::uniform_int_distribution< _IntType >::result_type**

The type of the range of the distribution.

Definition at line 1669 of file random.h.

4.999.3 Constructor & Destructor Documentation

4.999.3.1 template<typename _IntType = int> **std::uniform_int_distribution< _IntType >::uniform_int_distribution (_IntType *__a* = `0`, _IntType *__b* = std::numeric_limits<`_IntType`>`::`max`()`)** `[inline]`, `[explicit]`

Constructs a uniform distribution object.

Definition at line 1709 of file random.h.

4.999.4 Member Function Documentation

4.999.4.1 template<typename _IntType = int> **result_type std::uniform_int_distribution< _IntType >::max ( ) const** `[inline]`

Returns the inclusive upper bound of the distribution range.

Definition at line 1761 of file random.h.

4.999.4.2 template<typename _IntType = int> result_type std::uniform_int_distribution< _IntType >::min ( ) const [inline]

Returns the inclusive lower bound of the distribution range.

Definition at line 1754 of file random.h.

4.999.4.3 template<typename _IntType = int> template<typename _UniformRandomNumberGenerator > result_type std::uniform_int_distribution< _IntType >::operator() (_UniformRandomNumberGenerator & __urng) [inline]

Generating functions.

Definition at line 1769 of file random.h.

References std::uniform_int_distribution< _IntType >::operator()().

Referenced by std::uniform_int_distribution< _IntType >::operator()().

4.999.4.4 template<typename _IntType = int> param_type std::uniform_int_distribution< _IntType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 1739 of file random.h.

Referenced by std::operator>>().

4.999.4.5 template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::param (const param_type & __param) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 1747 of file random.h.

4.999.4.6 template<typename _IntType = int> void std::uniform_int_distribution< _IntType >::reset ( ) [inline]

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 1725 of file random.h.

4.999.5 Friends And Related Function Documentation

4.999.5.1 template<typename _IntType = int> bool operator== (const uniform_int_distribution< _IntType > & __d1, const uniform_int_distribution< _IntType > & __d2) [friend]

Return true if two uniform integer distributions have the same parameters.

Definition at line 1804 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.1000 std::uniform_int_distribution< _IntType >::param_type Struct Reference

Public Types

- typedef
  uniform_int_distribution
  < _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __a=0, _IntType __b=std::numeric_limits< _IntType >::max())
- result_type **a** () const
- result_type **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.1000.1 Detailed Description

template<typename _IntType = int>struct std::uniform_int_distribution< _IntType >::param_type

Parameter type.

Definition at line 1675 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.1001 std::uniform_real_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- uniform_real_distribution (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- **uniform_real_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type **a** () const
- result_type **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const uniform_real_distribution &__d1, const uniform_real_distribution &__d2)

4.1001.1 Detailed Description

template<typename _RealType = double>class std::uniform_real_distribution< _RealType >

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1867 of file random.h.

4.1001.2 Member Typedef Documentation

4.1001.2.1 template<typename _RealType = double> typedef _RealType std::uniform_real_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 1870 of file random.h.

4.1001.3 Constructor & Destructor Documentation

4.1001.3.1 template<typename _RealType = double> std::uniform_real_distribution< _RealType >::uniform_real_distribution (_RealType *__a* = `_RealType(0)`, _RealType *__b* = `_RealType(1)`) `[inline]`, `[explicit]`

Constructs a uniform_real_distribution object.

Parameters

__a	[IN] The lower bound of the distribution.
__b	[IN] The upper bound of the distribution.

Definition at line 1913 of file random.h.

4.1001.4 Member Function Documentation

4.1001.4.1 template<typename _RealType = double> result_type std::uniform_real_distribution< _RealType >::max ( ) const [inline]

Returns the inclusive upper bound of the distribution range.

Definition at line 1965 of file random.h.

4.1001.4.2 template<typename _RealType = double> result_type std::uniform_real_distribution< _RealType >::min ( ) const [inline]

Returns the inclusive lower bound of the distribution range.

Definition at line 1958 of file random.h.

4.1001.4.3 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::uniform_real_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 1973 of file random.h.

References std::uniform_real_distribution< _RealType >::operator()().

Referenced by std::uniform_real_distribution< _RealType >::operator()().

4.1001.4.4 template<typename _RealType = double> param_type std::uniform_real_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 1943 of file random.h.

Referenced by std::operator>>().

4.1001.4.5 template<typename _RealType = double> void std::uniform_real_distribution< _RealType >::param (const param_type & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 1951 of file random.h.

4.1001.4.6 template<typename _RealType = double> void std::uniform_real_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1929 of file random.h.

4.1001.5 Friends And Related Function Documentation

4.1001.5.1 template<typename _RealType = double> bool operator== (const **uniform_real_distribution**< _RealType > & *__d1,* const **uniform_real_distribution**< _RealType > & *__d2*) [friend]

Return true if two uniform real distributions have the same parameters.

Definition at line 2013 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.1002 std::uniform_real_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef
  uniform_real_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(0), _RealType __b=_RealType(1))
- result_type **a** () const
- result_type **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.1002.1 Detailed Description

template<typename _RealType = double>struct std::uniform_real_distribution< _RealType >::param_type

Parameter type.

Definition at line 1876 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.1003 std::unique_lock< _Mutex > Class Template Reference

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **unique_lock** (mutex_type &__m)
- **unique_lock** (mutex_type &__m, defer_lock_t) noexcept
- **unique_lock** (mutex_type &__m, try_to_lock_t)
- **unique_lock** (mutex_type &__m, adopt_lock_t)
- template<typename _Clock , typename _Duration >
  unique_lock (mutex_type &__m, const chrono::time_point< _Clock, _Duration > &__atime)
- template<typename _Rep , typename _Period >
  unique_lock (mutex_type &__m, const chrono::duration< _Rep, _Period > &__rtime)
- **unique_lock** (const unique_lock &)=delete
- **unique_lock** (unique_lock &&__u) noexcept
- void **lock** ()
- mutex_type ∗ **mutex** () const noexcept
- **operator bool** () const noexcept
- unique_lock & **operator=** (const unique_lock &)=delete
- unique_lock & **operator=** (unique_lock &&__u) noexcept
- bool **owns_lock** () const noexcept
- mutex_type ∗ **release** () noexcept
- void **swap** (unique_lock &__u) noexcept
- bool **try_lock** ()
- template<typename _Rep , typename _Period >
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock , typename _Duration >
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

4.1003.1 Detailed Description

template<typename _Mutex>class std::unique_lock< _Mutex >

unique_lock

Definition at line 429 of file mutex.

The documentation for this class was generated from the following file:

- mutex

4.1004 std::unique_ptr< _Tp, _Dp > Class Template Reference

Public Types

- typedef _Dp **deleter_type**
- typedef _Tp **element_type**
- typedef _Pointer::type **pointer**

Public Member Functions

- **unique_ptr** (pointer __p) noexcept
- **unique_ptr** (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) noexcept
- **unique_ptr** (pointer __p, typename remove_reference< deleter_type >::type &&__d) noexcept
- constexpr **unique_ptr** (nullptr_t) noexcept
- **unique_ptr** (unique_ptr &&__u) noexcept
- template<typename _Up , typename _Ep , typename = _Require< is_convertible<typename unique_ptr<_Up, _Ep>::pointer, pointer>, __not_<is_array<_Up>>, typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>::type>>
  unique_ptr (unique_ptr< _Up, _Ep > &&__u) noexcept
- **unique_ptr** (const unique_ptr &)=delete
- template<typename _Up , typename >
  unique_ptr (auto_ptr< _Up > &&__u) noexcept
- pointer **get** () const noexcept
- deleter_type & **get_deleter** () noexcept
- const deleter_type & **get_deleter** () const noexcept
- **operator bool** () const noexcept
- add_lvalue_reference
  < element_type >::type **operator*** () const
- pointer **operator->** () const noexcept
- unique_ptr & **operator=** (unique_ptr &&__u) noexcept
- template<typename _Up , typename _Ep >
  enable_if< __and_
  < is_convertible< typename
  unique_ptr< _Up, _Ep >
  ::pointer, pointer >, __not_
  < is_array< _Up > > >::value,
  unique_ptr & >::type **operator=** (unique_ptr< _Up, _Ep > &&__u) noexcept
- unique_ptr & **operator=** (nullptr_t) noexcept
- unique_ptr & **operator=** (const unique_ptr &)=delete
- pointer **release** () noexcept
- void **reset** (pointer __p=pointer()) noexcept
- void **swap** (unique_ptr &__u) noexcept

4.1004.1 Detailed Description

template<typename _Tp, typename _Dp = default_delete<_Tp>>class std::unique_ptr< _Tp, _Dp >

20.7.1.2 unique_ptr for single objects.

Definition at line 109 of file unique_ptr.h.

The documentation for this class was generated from the following files:

- unique_ptr.h
- auto_ptr.h

4.1005 std::unique_ptr< _Tp[], _Dp > Class Template Reference

Public Types

- typedef _Dp **deleter_type**
- typedef _Tp **element_type**
- typedef _Pointer::type **pointer**

Public Member Functions

- **unique_ptr** (pointer __p) noexcept
- template<typename _Up , typename = _Require<is_pointer<pointer>, is_convertible<_Up∗, pointer>, __is_derived_Tp<_Up>>> **unique_ptr** (_Up ∗__p)=delete
- **unique_ptr** (pointer __p, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type __d) noexcept
- **unique_ptr** (pointer __p, typename remove_reference< deleter_type >::type &&__d) noexcept
- **unique_ptr** (unique_ptr &&__u) noexcept
- constexpr **unique_ptr** (nullptr_t) noexcept
- template<typename _Up , typename _Ep , typename = _Require<__safe_conversion<_Up, _Ep>, typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>::type >> **unique_ptr** (unique_ptr< _Up, _Ep > &&__u) noexcept
- **unique_ptr** (const unique_ptr &)=delete
- template<typename _Up , typename = _Require<is_pointer<pointer>, is_convertible<_Up∗, pointer>, __is_derived_Tp<_Up>>> **unique_ptr** (_Up ∗, typename conditional< is_reference< deleter_type >::value, deleter_type, const deleter_type & >::type)=delete
- template<typename _Up , typename = _Require<is_pointer<pointer>, is_convertible<_Up∗, pointer>, __is_derived_Tp<_Up>>> **unique_ptr** (_Up ∗, typename remove_reference< deleter_type >::type &&)=delete
- pointer **get** () const noexcept
- deleter_type & **get_deleter** () noexcept
- const deleter_type & **get_deleter** () const noexcept
- **operator bool** () const noexcept
- unique_ptr & **operator=** (unique_ptr &&__u) noexcept
- template<typename _Up , typename _Ep > enable_if< __safe_conversion < _Up, _Ep >::value, unique_ptr & >::type **operator=** (unique_ptr< _Up, _Ep > &&__u) noexcept
- unique_ptr & **operator=** (nullptr_t) noexcept
- unique_ptr & **operator=** (const unique_ptr &)=delete
- std::add_lvalue_reference < element_type >::type **operator[ ]** (size_t __i) const
- pointer **release** () noexcept
- void **reset** () noexcept
- void **reset** (pointer __p) noexcept
- template<typename _Up , typename = _Require<is_pointer<pointer>, is_convertible<_Up∗, pointer>, __is_derived_Tp<_Up>>> void **reset** (_Up ∗)=delete
- void **swap** (unique_ptr &__u) noexcept

4.1005.1 Detailed Description

template<typename _Tp, typename _Dp>class std::unique_ptr< _Tp[ ], _Dp >

20.7.1.3 unique_ptr for array objects with a runtime length

Definition at line 282 of file unique_ptr.h.

The documentation for this class was generated from the following file:

- unique_ptr.h

4.1006 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Inherits std::__allow_copy_cons< bool >.

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::mapped_type mapped_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef allocator_type::pointer pointer
- typedef allocator_type::const_pointer const_pointer
- typedef allocator_type::reference reference
- typedef allocator_type::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_map (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator > unordered_map (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_map (const unordered_map &)=default
- unordered_map (unordered_map &&)=default
- unordered_map (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- iterator begin () noexcept

- local_iterator begin (size_type __n)
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- void clear () noexcept
- size_type count (const key_type &__x) const
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- local_iterator end (size_type __n)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_map & operator= (const unordered_map &)=default
- unordered_map & operator= (unordered_map &&)=default
- unordered_map & operator= (initializer_list< value_type > __l)
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_map &__x)

- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept

- const_iterator end () const noexcept
- const_iterator cend () const noexcept

- std::pair< iterator, bool > insert (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > insert (_Pair &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator insert (const_iterator __hint, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __it)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator,
  const_iterator > equal_range (const key_type &__x) const

- mapped_type & operator[] (const key_type &__k)
- mapped_type & operator[] (key_type &&__k)

- mapped_type & at (const key_type &__k)
- const mapped_type & at (const key_type &__k) const

- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Key1 , typename _Tp1 , typename _Hash1 , typename _Pred1 , typename _Alloc1 >
  bool **operator==** (const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)

4.1006.1 Detailed Description

template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >>class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is std::pair<const _Key, _Tp>.

Base is _Hashtable, dispatched at compile time via template alias __umap_hashtable.

Definition at line 97 of file unordered_map.h.

4.1006.2 Member Typedef Documentation

4.1006.2.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type

Public typedefs.

Definition at line 111 of file unordered_map.h.

4.1006.2.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator

Iterator-related typedefs.

Definition at line 121 of file unordered_map.h.

4.1006.2.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator

Iterator-related typedefs.

Definition at line 123 of file unordered_map.h.

4.1006.2.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef allocator_type::const_pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer

Iterator-related typedefs.

Definition at line 117 of file unordered_map.h.

4.1006.2.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef allocator_type::const_reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference

Iterator-related typedefs.

Definition at line 119 of file unordered_map.h.

4.1006.2.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::difference_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type

Iterator-related typedefs.

Definition at line 125 of file unordered_map.h.

4.1006.2.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher

Public typedefs.

Definition at line 109 of file unordered_map.h.

4.1006.2.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator

Iterator-related typedefs.

Definition at line 120 of file unordered_map.h.

4.1006.2.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal

Public typedefs.

Definition at line 110 of file unordered_map.h.

4.1006.2.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type

Public typedefs.

Definition at line 106 of file unordered_map.h.

4.1006.2.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator

Iterator-related typedefs.

Definition at line 122 of file unordered_map.h.

4.1006.2.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::mapped_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type

Public typedefs.

Definition at line 108 of file unordered_map.h.

4.1006.2.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer

Iterator-related typedefs.

Definition at line 116 of file unordered_map.h.

4.1006.2.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reference

Iterator-related typedefs.

Definition at line 118 of file unordered_map.h.

4.1006.2.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::size_type **std::unordered_map**< _Key, _Tp, _Hash, _Pred, _Alloc >::**size_type**

Iterator-related typedefs.

Definition at line 124 of file unordered_map.h.

4.1006.2.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > typedef _Hashtable::value_type **std::unordered_map**< _Key, _Tp, _Hash, _Pred, _Alloc >::**value_type**

Public typedefs.

Definition at line 107 of file unordered_map.h.

4.1006.3 Constructor & Destructor Documentation

4.1006.3.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > **std::unordered_map**< _Key, _Tp, _Hash, _Pred, _Alloc >::**unordered_map** (**size_type** *__n* = `10`, const **hasher** & *__hf* = **hasher** `()`, const **key_equal** & *__eql* = **key_equal** `()`, const **allocator_type** & *__a* = **allocator_type** `()`) `[inline]`, `[explicit]`

Default constructor creates no elements.

Parameters

__n	Initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Definition at line 138 of file unordered_map.h.

4.1006.3.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> > > template<typename _InputIterator > **std::unordered_map**< _Key, _Tp, _Hash, _Pred, _Alloc >::**unordered_map** (_InputIterator *__f*, _InputIterator *__l*, **size_type** *__n* = `0`, const **hasher** & *__hf* = **hasher** `()`, const **key_equal** & *__eql* = **key_equal** `()`, const **allocator_type** & *__a* = **allocator_type** `()`) `[inline]`

Builds an unordered_map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

Definition at line 159 of file unordered_map.h.

4.1006.3.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]

Copy constructor.

4.1006.3.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]

Move constructor.

4.1006.3.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eql = key_equal(), const allocator_type & __a = allocator_type()) [inline]

Builds an unordered_map from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

Definition at line 184 of file unordered_map.h.

4.1006.4 Member Function Documentation

4.1006.4.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (const key_type & __k) [inline]

Access to unordered_map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to *__k*, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

Definition at line 612 of file unordered_map.h.

4.1006.4.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (const key_type & __k) const [inline]

Access to unordered_map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to *__k*, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

Definition at line 616 of file unordered_map.h.

4.1006.4.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first element in the unordered_map.

Definition at line 248 of file unordered_map.h.

4.1006.4.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

Definition at line 257 of file unordered_map.h.

4.1006.4.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (size_type __n) [inline]

Returns a read/write iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read/write local iterator.

Definition at line 657 of file unordered_map.h.

4.1006.4.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 668 of file unordered_map.h.

4.1006.4.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline], [noexcept]

Returns the number of buckets of the unordered_map.

Definition at line 624 of file unordered_map.h.

4.1006.4.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

Definition at line 261 of file unordered_map.h.

4.1006.4.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 672 of file unordered_map.h.

4.1006.4.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

Definition at line 283 of file unordered_map.h.

4.1006.4.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 698 of file unordered_map.h.

4.1006.4.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in an unordered_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 497 of file unordered_map.h.

4.1006.4.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (const key_type & *__x*) const [inline]

Finds the number of elements.

Parameters

__x	Key to count.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multimap; for unordered_map the result will either be 0 (not present) or 1 (present).

Definition at line 560 of file unordered_map.h.

4.1006.4.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (_Args &&... *__args*) [inline]

Attempts to build and insert a std::pair into the unordered_map.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.

Insertion requires amortized constant time.

Definition at line 310 of file unordered_map.h.

4.1006.4.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (const_iterator __pos, _Args &&... __args) [inline]

Attempts to build and insert a std::pair into the unordered_map.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args* (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 340 of file unordered_map.h.

4.1006.4.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the unordered_map is empty.

Definition at line 228 of file unordered_map.h.

4.1006.4.17 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last element in the unordered_map.

Definition at line 270 of file unordered_map.h.

4.1006.4.18 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

Definition at line 279 of file unordered_map.h.

4.1006.4.19 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (size_type __n) [inline]

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read/write local iterator.

Definition at line 683 of file unordered_map.h.

4.1006.4.20 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 694 of file unordered_map.h.

4.1006.4.21 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, iterator> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & *__x*) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

Definition at line 573 of file unordered_map.h.

4.1006.4.22 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & *__x*) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

Definition at line 577 of file unordered_map.h.

4.1006.4.23 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (const_iterator *__position*) [inline]

Erases an element from an unordered_map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 447 of file unordered_map.h.

4.1006.4.24 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (iterator *__it*) [inline]

Erases an element from an unordered_map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 452 of file unordered_map.h.

4.1006.4.25 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (const key_type & *__x*) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_map. For an unordered_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 469 of file unordered_map.h.

4.1006.4.26 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (const_iterator *__first*, const_iterator *__last*) [inline]

Erases a [__first,__last) range of elements from an unordered_map.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_map. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 487 of file unordered_map.h.

4.1006.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (const key_type & *__x*) [inline]

Tries to locate an element in an unordered_map.

Parameters

__x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 542 of file unordered_map.h.

4.1006.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (const key_type & __x) const [inline]

Tries to locate an element in an unordered_map.

Parameters

__x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 546 of file unordered_map.h.

4.1006.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Returns the allocator object with which the unordered_map was constructed.

Definition at line 221 of file unordered_map.h.

4.1006.4.30 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]

Returns the hash functor object with which the unordered_map was constructed.

Definition at line 518 of file unordered_map.h.

4.1006.4.31 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (const value_type & __x) [inline]

Attempts to insert a std::pair into the unordered_map.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.

Insertion requires amortized constant time.

Definition at line 362 of file unordered_map.h.

4.1006.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (_Pair && __x) [inline]

Attempts to insert a std::pair into the unordered_map.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.

Insertion requires amortized constant time.

Definition at line 369 of file unordered_map.h.

4.1006.4.33 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (const_iterator *__hint*, const value_type & *__x*) [inline]

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 396 of file unordered_map.h.

4.1006.4.34 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (const_iterator __*hint*, _Pair && __*x*) [inline]

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 403 of file unordered_map.h.

4.1006.4.35 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (_InputIterator __*first*, _InputIterator __*last*) [inline]

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 418 of file unordered_map.h.

4.1006.4.36 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (initializer_list< value_type > __*l*) [inline]

Attempts to insert a list of elements into the unordered_map.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 429 of file unordered_map.h.

4.1006.4.37 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]

Returns the key comparison object with which the unordered_map was constructed.

Definition at line 524 of file unordered_map.h.

4.1006.4.38 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]

Returns the average number of elements per bucket.

Definition at line 706 of file unordered_map.h.

4.1006.4.39 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline], [noexcept]

Returns the maximum number of buckets of the unordered_map.

Definition at line 629 of file unordered_map.h.

4.1006.4.40 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline], [noexcept]

Returns a positive number that the unordered_map tries to keep the load factor less than or equal to.

Definition at line 712 of file unordered_map.h.

4.1006.4.41 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (float *__z*) [inline]

Change the unordered_map maximum load factor.

Parameters

__z	The new maximum load factor.

Definition at line 720 of file unordered_map.h.

4.1006.4.42 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the unordered_map.

Definition at line 238 of file unordered_map.h.

4.1006.4.43 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]

Copy assignment operator.

4.1006.4.44 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]

Move assignment operator.

4.1006.4.45 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Unordered_map list assignment operator.

Parameters

__l	An initializer_list.

This function fills an unordered_map with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the unordered_map and that the resulting unordered_map's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 212 of file unordered_map.h.

4.1006.4.46 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (const key_type & __k) [inline]

Subscript ([]) access to unordered_map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 595 of file unordered_map.h.

4.1006.4.47 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> mapped_type& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (key_type && __k) [inline]

Subscript ([]) access to unordered_map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 599 of file unordered_map.h.

4.1006.4.48 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (size_type __n) [inline]

May rehash the unordered_map.

Parameters

__n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_map maximum load factor.

Definition at line 731 of file unordered_map.h.

4.1006.4.49 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (size_type __n) [inline]

Prepare the unordered_map for a specified number of elements.

Parameters

__n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

Definition at line 742 of file unordered_map.h.

4.1006.4.50 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the unordered_map.

Definition at line 233 of file unordered_map.h.

4.1006.4.51 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline]

Swaps data with another unordered_map.

Parameters

__x	An unordered_map of the same element and allocator types.

This exchanges the elements between two unordered_map in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 510 of file unordered_map.h.

The documentation for this class was generated from the following file:

- unordered_map.h

4.1007 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

Inherits std::__allow_copy_cons< bool >.

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::mapped_type mapped_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef allocator_type::pointer pointer
- typedef allocator_type::const_pointer const_pointer
- typedef allocator_type::reference reference
- typedef allocator_type::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_multimap (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator > unordered_multimap (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multimap (const unordered_multimap &)=default
- unordered_multimap (unordered_multimap &&)=default
- unordered_multimap (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- iterator begin () noexcept
- local_iterator begin (size_type __n)
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- void clear () noexcept

- size_type count (const key_type &__x) const
- template<typename... _Args>
  iterator emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- local_iterator end (size_type __n)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_multimap & operator= (const unordered_multimap &)=default
- unordered_multimap & operator= (unordered_multimap &&)=default
- unordered_multimap & operator= (initializer_list< value_type > __l)
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_multimap &__x)

- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept

- const_iterator end () const noexcept
- const_iterator cend () const noexcept

- iterator insert (const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator insert (_Pair &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator insert (const_iterator __hint, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __it)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)

- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Key1 , typename _Tp1 , typename _Hash1 , typename _Pred1 , typename _Alloc1 >
  bool **operator==** (const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)

4.1007.1 Detailed Description

template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >>class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is std::pair<const _Key, _Tp>.

Base is _Hashtable, dispatched at compile time via template alias __ummap_hashtable.

Definition at line 778 of file unordered_map.h.

4.1007.2 Member Typedef Documentation

4.1007.2.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::allocator_type **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type**

Public typedefs.

Definition at line 792 of file unordered_map.h.

4.1007.2.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::const_iterator **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator**

Iterator-related typedefs.

Definition at line 802 of file unordered_map.h.

4.1007.2.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::const_local_iterator **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator**

Iterator-related typedefs.

Definition at line 804 of file unordered_map.h.

4.1007.2.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::const_pointer **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer**

Iterator-related typedefs.

Definition at line 798 of file unordered_map.h.

4.1007.2.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::const_reference **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference**

Iterator-related typedefs.

Definition at line 800 of file unordered_map.h.

4.1007.2.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::difference_type **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type**

Iterator-related typedefs.

Definition at line 806 of file unordered_map.h.

4.1007.2.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::hasher **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher**

Public typedefs.

Definition at line 790 of file unordered_map.h.

4.1007.2.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::iterator **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator**

Iterator-related typedefs.

Definition at line 801 of file unordered_map.h.

4.1007.2.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_equal **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal**

Public typedefs.

Definition at line 791 of file unordered_map.h.

4.1007.2.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::key_type **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type**

Public typedefs.

Definition at line 787 of file unordered_map.h.

4.1007.2.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::local_iterator **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator**

Iterator-related typedefs.

Definition at line 803 of file unordered_map.h.

4.1007.2.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::mapped_type **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type**

Public typedefs.

Definition at line 789 of file unordered_map.h.

4.1007.2.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::pointer **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer**

Iterator-related typedefs.

Definition at line 797 of file unordered_map.h.

4.1007.2.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef allocator_type::reference **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference**

Iterator-related typedefs.

Definition at line 799 of file unordered_map.h.

4.1007.2.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::size_type **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type**

Iterator-related typedefs.

Definition at line 805 of file unordered_map.h.

4.1007.2.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> typedef _Hashtable::value_type **std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type**

Public typedefs.

Definition at line 788 of file unordered_map.h.

4.1007.3 Constructor & Destructor Documentation

4.1007.3.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (size_type *__n* = 10, const hasher & *__hf* = hasher(), const key_equal & *__eql* = key_equal(), const allocator_type & *__a* = allocator_type()) [inline], [explicit]

Default constructor creates no elements.

Parameters

__n	Initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Definition at line 819 of file unordered_map.h.

4.1007.3.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (_InputIterator *__f*, _InputIterator *__l*, size_type *__n* = 0, const hasher & *__hf* = hasher(), const key_equal & *__eql* = key_equal(), const allocator_type & *__a* = allocator_type()) [inline]

Builds an unordered_multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multimap consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

Definition at line 840 of file unordered_map.h.

4.1007.3.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]

Copy constructor.

4.1007.3.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]

Move constructor.

4.1007.3.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eql = key_equal(), const allocator_type & __a = allocator_type()) [inline]

Builds an unordered_multimap from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multimap consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

Definition at line 865 of file unordered_map.h.

4.1007.4 Member Function Documentation

4.1007.4.1 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first element in the unordered_multimap.

Definition at line 929 of file unordered_map.h.

4.1007.4.2 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

Definition at line 938 of file unordered_map.h.

4.1007.4.3 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (size_type __n) [inline]

Returns a read/write iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read/write local iterator.

Definition at line 1277 of file unordered_map.h.

4.1007.4.4 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1288 of file unordered_map.h.

4.1007.4.5 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline], [noexcept]

Returns the number of buckets of the unordered_multimap.

Definition at line 1244 of file unordered_map.h.

4.1007.4.6 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

Definition at line 942 of file unordered_map.h.

4.1007.4.7 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1292 of file unordered_map.h.

4.1007.4.8 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

Definition at line 964 of file unordered_map.h.

4.1007.4.9 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1318 of file unordered_map.h.

4.1007.4.10 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in an unordered_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1161 of file unordered_map.h.

4.1007.4.11 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (const key_type & __x) const [inline]

Finds the number of elements.

Parameters

__x	Key to count.

Returns

Number of elements with specified key.

Definition at line 1221 of file unordered_map.h.

4.1007.4.12 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (_Args &&... __args) [inline]

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the unordered_multimap.

Insertion requires amortized constant time.

Definition at line 987 of file unordered_map.h.

4.1007.4.13 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename... _Args> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (const_iterator __pos, _Args &&... __args) [inline]

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args*.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1013 of file unordered_map.h.

4.1007.4.14 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the unordered_multimap is empty.

Definition at line 909 of file unordered_map.h.

4.1007.4.15 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]

Returns a read/write iterator that points one past the last element in the unordered_multimap.

Definition at line 951 of file unordered_map.h.

4.1007.4.16 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

Definition at line 960 of file unordered_map.h.

4.1007.4.17 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (size_type __n) [inline]

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read/write local iterator.

Definition at line 1303 of file unordered_map.h.

4.1007.4.18 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1314 of file unordered_map.h.

4.1007.4.19 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<iterator, iterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & __x) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1232 of file unordered_map.h.

4.1007.4.20 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> std::pair<const_iterator, const_iterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (const key_type & *__x*) const `[inline]`

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1236 of file unordered_map.h.

4.1007.4.21 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (const_iterator *__position*) `[inline]`

Erases an element from an unordered_multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1111 of file unordered_map.h.

4.1007.4.22 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (iterator *__it*) `[inline]`

Erases an element from an unordered_multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1116 of file unordered_map.h.

4.1007.4.23 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (const key_type & *__x*) `[inline]`

Erases elements according to the provided key.

Parameters

__x	Key of elements to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1132 of file unordered_map.h.

4.1007.4.24 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (const_iterator *__first*, const_iterator *__last*) `[inline]`

Erases a [__first,__last) range of elements from an unordered_multimap.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_multimap. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1151 of file unordered_map.h.

4.1007.4.25 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (const key_type & __x) [inline]

Tries to locate an element in an unordered_multimap.

Parameters

__x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1207 of file unordered_map.h.

4.1007.4.26 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (const key_type & __x) const [inline]

Tries to locate an element in an unordered_multimap.

Parameters

__x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1211 of file unordered_map.h.

4.1007.4.27 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Returns the allocator object with which the unordered_multimap was constructed.

Definition at line 902 of file unordered_map.h.

4.1007.4.28 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]

Returns the hash functor object with which the unordered_multimap was constructed.

Definition at line 1183 of file unordered_map.h.

4.1007.4.29 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (const value_type & __x) [inline]

Inserts a std::pair into the unordered_multimap.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1027 of file unordered_map.h.

4.1007.4.30 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (_Pair && __x) [inline]

Inserts a std::pair into the unordered_multimap.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1034 of file unordered_map.h.

4.1007.4.31 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (const_iterator __hint, const value_type & __x) [inline]

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1059 of file unordered_map.h.

4.1007.4.32 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _Pair , typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (const_iterator *__hint,* _Pair && *__x*) `[inline]`

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html` for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1066 of file unordered_map.h.

4.1007.4.33 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> template<typename _InputIterator > void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (_InputIterator *__first,* _InputIterator *__last*) `[inline]`

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 1081 of file unordered_map.h.

4.1007.4.34 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (initializer_list< value_type > *__l*) `[inline]`

Attempts to insert a list of elements into the unordered_multimap.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 1093 of file unordered_map.h.

4.1007.4.35 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]

Returns the key comparison object with which the unordered_multimap was constructed.

Definition at line 1189 of file unordered_map.h.

4.1007.4.36 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]

Returns the average number of elements per bucket.

Definition at line 1326 of file unordered_map.h.

4.1007.4.37 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline], [noexcept]

Returns the maximum number of buckets of the unordered_multimap.

Definition at line 1249 of file unordered_map.h.

4.1007.4.38 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline], [noexcept]

Returns a positive number that the unordered_multimap tries to keep the load factor less than or equal to.

Definition at line 1332 of file unordered_map.h.

4.1007.4.39 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (float __z) [inline]

Change the unordered_multimap maximum load factor.

Parameters

__z	The new maximum load factor.

Definition at line 1340 of file unordered_map.h.

4.1007.4.40 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the unordered_multimap.

Definition at line 919 of file unordered_map.h.

4.1007.4.41 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &) `[default]`

Copy assignment operator.

4.1007.4.42 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &&) `[default]`

Move assignment operator.

4.1007.4.43 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (initializer_list< value_type > *__l*) `[inline]`

Unordered_multimap list assignment operator.

Parameters

__l	An initializer_list.

This function fills an unordered_multimap with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the unordered_multimap and that the resulting unordered_multimap's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 893 of file unordered_map.h.

4.1007.4.44 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (size_type *__n*) `[inline]`

May rehash the unordered_multimap.

Parameters

__n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_multimap maximum load factor.

Definition at line 1351 of file unordered_map.h.

4.1007.4.45 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (size_type *__n*) `[inline]`

Prepare the unordered_multimap for a specified number of elements.

Parameters

__n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

Definition at line 1362 of file unordered_map.h.

4.1007.4.46 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the unordered_multimap.

Definition at line 914 of file unordered_map.h.

4.1007.4.47 template<class _Key, class _Tp, class _Hash = hash<_Key>, class _Pred = std::equal_to<_Key>, class _Alloc = std::allocator<std::pair<const _Key, _Tp> >> void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline]

Swaps data with another unordered_multimap.

Parameters

__x	An unordered_multimap of the same element and allocator types.

This exchanges the elements between two unordered_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.

Definition at line 1175 of file unordered_map.h.

The documentation for this class was generated from the following file:

- unordered_map.h

4.1008 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

Inherits std::__allow_copy_cons< bool >.

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef allocator_type::pointer pointer
- typedef allocator_type::const_pointer const_pointer
- typedef allocator_type::reference reference
- typedef allocator_type::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_multiset (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_multiset (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multiset (const unordered_multiset &)=default
- unordered_multiset (unordered_multiset &&)=default
- unordered_multiset (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- size_type count (const key_type &__x) const
- template<typename... _Args>
  iterator emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_multiset & operator= (const unordered_multiset &)=default
- unordered_multiset & operator= (unordered_multiset &&__x)=default
- unordered_multiset & operator= (initializer_list< value_type > __l)
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_multiset &__x)

- iterator begin () noexcept
- const_iterator begin () const noexcept

- iterator end () noexcept
- const_iterator end () const noexcept

- iterator insert (const value_type &__x)

- iterator insert (value_type &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __it)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

- local_iterator begin (size_type __n)
- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- local_iterator end (size_type __n)
- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Value1 , typename _Hash1 , typename _Pred1 , typename _Alloc1 >
  bool **operator==** (const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &)

4.1008.1 Detailed Description

template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Template Parameters

_Value	Type of key objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a `container`, and `unordered associative container`

Base is _Hashtable, dispatched at compile time via template alias __umset_hashtable.

Definition at line 698 of file unordered_set.h.

4.1008.2 Member Typedef Documentation

4.1008.2.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**allocator_type**

Public typedefs.

Definition at line 711 of file unordered_set.h.

4.1008.2.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**const_iterator**

Iterator-related typedefs.

Definition at line 721 of file unordered_set.h.

4.1008.2.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**const_local_iterator**

Iterator-related typedefs.

Definition at line 723 of file unordered_set.h.

4.1008.2.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_pointer **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**const_pointer**

Iterator-related typedefs.

Definition at line 717 of file unordered_set.h.

4.1008.2.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_reference **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**const_reference**

Iterator-related typedefs.

Definition at line 719 of file unordered_set.h.

4.1008.2.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::difference_type **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**difference_type**

Iterator-related typedefs.

Definition at line 725 of file unordered_set.h.

4.1008.2.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::hasher **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::**hasher**

Public typedefs.

Definition at line 709 of file unordered_set.h.

4.1008.2.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::iterator

Iterator-related typedefs.

Definition at line 720 of file unordered_set.h.

4.1008.2.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal

Public typedefs.

Definition at line 710 of file unordered_set.h.

4.1008.2.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type

Public typedefs.

Definition at line 707 of file unordered_set.h.

4.1008.2.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::local_iterator

Iterator-related typedefs.

Definition at line 722 of file unordered_set.h.

4.1008.2.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer

Iterator-related typedefs.

Definition at line 716 of file unordered_set.h.

4.1008.2.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference

Iterator-related typedefs.

Definition at line 718 of file unordered_set.h.

4.1008.2.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type

Iterator-related typedefs.

Definition at line 724 of file unordered_set.h.

4.1008.2.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::value_type **std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_type**

Public typedefs.

Definition at line 708 of file unordered_set.h.

4.1008.3 Constructor & Destructor Documentation

4.1008.3.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> **std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (size_type** *__n* = `10`, const **hasher** & *__hf* = **hasher** `()`, const **key_equal** & *__eql* = **key_equal** `()`, const **allocator_type** & *__a* = **allocator_type** `()` **)** `[inline], [explicit]`

Default constructor creates no elements.

Parameters

__n	Initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Definition at line 737 of file unordered_set.h.

4.1008.3.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename _InputIterator > **std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (** _InputIterator *__f*, _InputIterator *__l*, **size_type** *__n* = `0`, const **hasher** & *__hf* = **hasher** `()`, const **key_equal** & *__eql* = **key_equal** `()`, const **allocator_type** & *__a* = **allocator_type** `()` **)** `[inline]`

Builds an unordered_multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multiset consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

Definition at line 758 of file unordered_set.h.

4.1008.3.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> **std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (** const **unordered_multiset**< _Value, _Hash, _Pred, _Alloc > & **)** `[default]`

Copy constructor.

4.1008.3.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &&) [default]

Move constructor.

4.1008.3.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (initializer_list< value_type > __l, size_type __n = 0, const hasher & __hf = hasher(), const key_equal & __eql = key_equal(), const allocator_type & __a = allocator_type()) [inline]

Builds an unordered_multiset from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multiset consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

Definition at line 783 of file unordered_set.h.

4.1008.4 Member Function Documentation

4.1008.4.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

Definition at line 848 of file unordered_set.h.

4.1008.4.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

Definition at line 852 of file unordered_set.h.

4.1008.4.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (size_type __n) [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1175 of file unordered_set.h.

4.1008.4.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1179 of file unordered_set.h.

4.1008.4.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline], [noexcept]

Returns the number of buckets of the unordered_multiset.

Definition at line 1141 of file unordered_set.h.

4.1008.4.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

Definition at line 875 of file unordered_set.h.

4.1008.4.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1183 of file unordered_set.h.

4.1008.4.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

Definition at line 883 of file unordered_set.h.

4.1008.4.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (size_type __n) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1203 of file unordered_set.h.

4.1008.4.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in an unordered_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1059 of file unordered_set.h.

4.1008.4.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (const key_type & __x) const [inline]

Finds the number of elements.

Parameters

__x	Element to located.

Returns

Number of elements with specified key.

Definition at line 1118 of file unordered_set.h.

4.1008.4.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (_Args &&... __args) [inline]

Builds and insert an element into the unordered_multiset.

Parameters

__args	Arguments used to generate an element.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 897 of file unordered_set.h.

4.1008.4.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (const_iterator __pos, _Args &&... __args) [inline]

Inserts an element into the unordered_multiset.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html

Insertion requires amortized constant time.

Definition at line 919 of file unordered_set.h.

4.1008.4.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the unordered_multiset is empty.

Definition at line 827 of file unordered_set.h.

4.1008.4.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

Definition at line 862 of file unordered_set.h.

4.1008.4.16 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

Definition at line 866 of file unordered_set.h.

4.1008.4.17 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (size_type *__n*) [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1195 of file unordered_set.h.

4.1008.4.18 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 1199 of file unordered_set.h.

4.1008.4.19 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_range (const key_type & *__x*) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1129 of file unordered_set.h.

4.1008.4.20 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_range (const key_type & __x) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1133 of file unordered_set.h.

4.1008.4.21 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (const_iterator __position) [inline]

Erases an element from an unordered_multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1005 of file unordered_set.h.

4.1008.4.22 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (iterator __it) [inline]

Erases an element from an unordered_multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1010 of file unordered_set.h.

4.1008.4.23 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (const key_type & *__x*) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1028 of file unordered_set.h.

4.1008.4.24 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (const_iterator *__first*, const_iterator *__last*) [inline]

Erases a [__first,__last) range of elements from an unordered_multiset.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1048 of file unordered_set.h.

4.1008.4.25 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (const key_type & *__x*) [inline]

Tries to locate an element in an unordered_multiset.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1104 of file unordered_set.h.

4.1008.4.26 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (const key_type & __x) const [inline]

Tries to locate an element in an unordered_multiset.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1108 of file unordered_set.h.

4.1008.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Returns the allocator object with which the unordered_multiset was constructed.

Definition at line 820 of file unordered_set.h.

4.1008.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]

Returns the hash functor object with which the unordered_multiset was constructed.

Definition at line 1080 of file unordered_set.h.

4.1008.4.29 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (const value_type & __x) [inline]

Inserts an element into the unordered_multiset.

Parameters

__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 931 of file unordered_set.h.

4.1008.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (value_type && __x) [inline]

Inserts an element into the unordered_multiset.

Parameters

__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 935 of file unordered_set.h.

4.1008.4.31 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (const_iterator __hint, const value_type & __x) [inline]

Inserts an element into the unordered_multiset.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html

Insertion requires amortized constant.

Definition at line 957 of file unordered_set.h.

4.1008.4.32 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (const_iterator __hint, value_type && __x) [inline]

Inserts an element into the unordered_multiset.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html`

Insertion requires amortized constant.

Definition at line 961 of file unordered_set.h.

4.1008.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename _InputIterator > void **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::insert (_InputIterator *__first,* _InputIterator *__last*) `[inline]`

A template function that inserts a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 975 of file unordered_set.h.

4.1008.4.34 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void **std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::insert (**initializer_list**< **value_type** > *__l*) `[inline]`

Inserts a list of elements into the unordered_multiset.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 986 of file unordered_set.h.

4.1008.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> **key_equal std::unordered_multiset**< _Value, _Hash, _Pred, _Alloc >::key_eq ( ) const `[inline]`

Returns the key comparison object with which the unordered_multiset was constructed.

Definition at line 1086 of file unordered_set.h.

4.1008.4.36 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]

Returns the average number of elements per bucket.

Definition at line 1211 of file unordered_set.h.

4.1008.4.37 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline], [noexcept]

Returns the maximum number of buckets of the unordered_multiset.

Definition at line 1146 of file unordered_set.h.

4.1008.4.38 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline], [noexcept]

Returns a positive number that the unordered_multiset tries to keep the load factor less than or equal to.

Definition at line 1217 of file unordered_set.h.

4.1008.4.39 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (float __z) [inline]

Change the unordered_multiset maximum load factor.

Parameters

__z	The new maximum load factor.

Definition at line 1225 of file unordered_set.h.

4.1008.4.40 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size ( ) const [inline], [noexcept]

Returns the maximum size of the unordered_multiset.

Definition at line 837 of file unordered_set.h.

4.1008.4.41 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &) [default]

Copy assignment operator.

4.1008.4.42 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (unordered_multiset< _Value, _Hash, _Pred, _Alloc > && __x) [default]

Move assignment operator.

4.1008.4.43 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Unordered_multiset list assignment operator.

Parameters

__l	An initializer_list.

This function fills an unordered_multiset with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the unordered_multiset and that the resulting unordered_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 811 of file unordered_set.h.

4.1008.4.44 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (size_type __n) [inline]

May rehash the unordered_multiset.

Parameters

__n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_multiset maximum load factor.

Definition at line 1236 of file unordered_set.h.

4.1008.4.45 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (size_type __n) [inline]

Prepare the unordered_multiset for a specified number of elements.

Parameters

__n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

Definition at line 1247 of file unordered_set.h.

4.1008.4.46 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the unordered_multiset.

Definition at line 832 of file unordered_set.h.

4.1008.4.47 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x) [inline]

Swaps data with another unordered_multiset.

Parameters

__x	An unordered_multiset of the same element and allocator types.

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

Definition at line 1072 of file unordered_set.h.

The documentation for this class was generated from the following file:

- unordered_set.h

4.1009 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

Inherits std::__allow_copy_cons< bool >.

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef allocator_type::pointer pointer
- typedef allocator_type::const_pointer const_pointer
- typedef allocator_type::reference reference
- typedef allocator_type::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_set (size_type __n=10, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- template<typename _InputIterator >
  unordered_set (_InputIterator __f, _InputIterator __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_set (const unordered_set &)=default
- unordered_set (unordered_set &&)=default
- unordered_set (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept

- size_type **bucket_size** (size_type __n) const
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- size_type count (const key_type &__x) const
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&...__args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&...__args)
- bool empty () const noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator >
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_set & operator= (const unordered_set &)=default
- unordered_set & operator= (unordered_set &&)=default
- unordered_set & operator= (initializer_list< value_type > __l)
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_set &__x)

- iterator begin () noexcept
- const_iterator begin () const noexcept

- iterator end () noexcept
- const_iterator end () const noexcept

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __it)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)

- std::pair< const_iterator,
  const_iterator > equal_range (const key_type &__x) const

- local_iterator begin (size_type __n)
- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- local_iterator end (size_type __n)
- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Value1 , typename _Hash1 , typename _Pred1 , typename _Alloc1 >
  bool **operator==** (const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &)

4.1009.1 Detailed Description

template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>>class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Template Parameters

_Value	Type of key objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a `container`, and `unordered associative container`

Base is _Hashtable, dispatched at compile time via template alias __uset_hashtable.

Definition at line 93 of file unordered_set.h.

4.1009.2 Member Typedef Documentation

4.1009.2.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::allocator_type

Public typedefs.

Definition at line 106 of file unordered_set.h.

4.1009.2.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_iterator

Iterator-related typedefs.

Definition at line 116 of file unordered_set.h.

4.1009.2.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_local_iterator

Iterator-related typedefs.

Definition at line 118 of file unordered_set.h.

4.1009.2.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_pointer

Iterator-related typedefs.

Definition at line 112 of file unordered_set.h.

4.1009.2.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::const_reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_reference

Iterator-related typedefs.

Definition at line 114 of file unordered_set.h.

4.1009.2.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::difference_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::difference_type

Iterator-related typedefs.

Definition at line 120 of file unordered_set.h.

4.1009.2.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hasher

Public typedefs.

Definition at line 104 of file unordered_set.h.

4.1009.2.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::iterator

Iterator-related typedefs.

Definition at line 115 of file unordered_set.h.

4.1009.2.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_equal

Public typedefs.

Definition at line 105 of file unordered_set.h.

4.1009.2.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::key_type **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::key_type**

Public typedefs.

Definition at line 102 of file unordered_set.h.

4.1009.2.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::local_iterator **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::local_iterator**

Iterator-related typedefs.

Definition at line 117 of file unordered_set.h.

4.1009.2.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::pointer **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::pointer**

Iterator-related typedefs.

Definition at line 111 of file unordered_set.h.

4.1009.2.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef allocator_type::reference **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::reference**

Iterator-related typedefs.

Definition at line 113 of file unordered_set.h.

4.1009.2.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::size_type **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::size_type**

Iterator-related typedefs.

Definition at line 119 of file unordered_set.h.

4.1009.2.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> typedef _Hashtable::value_type **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::value_type**

Public typedefs.

Definition at line 103 of file unordered_set.h.

4.1009.3 Constructor & Destructor Documentation

4.1009.3.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >**::unordered_set** (**size_type** *__n* = `10`, const **hasher** & *__hf* = **hasher** `()`, const **key_equal** & *__eql* = **key_equal** `()`, const **allocator_type** & *__a* = **allocator_type** `()`) `[inline]`, `[explicit]`

Default constructor creates no elements.

Parameters

__n	Initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Definition at line 132 of file unordered_set.h.

4.1009.3.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename _InputIterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (_InputIterator *__f*, _InputIterator *__l*, size_type *__n* = `0`, const hasher & *__hf* = hasher `()`, const key_equal & *__eql* = key_equal `()`, const allocator_type & *__a* = allocator_type `()`) `[inline]`

Builds an unordered_set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_set consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

Definition at line 153 of file unordered_set.h.

4.1009.3.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (const unordered_set< _Value, _Hash, _Pred, _Alloc > &) `[default]`

Copy constructor.

4.1009.3.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (unordered_set< _Value, _Hash, _Pred, _Alloc > &&) `[default]`

Move constructor.

4.1009.3.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (initializer_list< value_type > *__l*, size_type *__n* = `0`, const hasher & *__hf* = hasher `()`, const key_equal & *__eql* = key_equal `()`, const allocator_type & *__a* = allocator_type `()`) `[inline]`

Builds an unordered_set from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_set consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

Definition at line 178 of file unordered_set.h.

4.1009.4 Member Function Documentation

4.1009.4.1 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

Definition at line 243 of file unordered_set.h.

4.1009.4.2 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

Definition at line 247 of file unordered_set.h.

4.1009.4.3 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (size_type *__n*) [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 593 of file unordered_set.h.

4.1009.4.4 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 597 of file unordered_set.h.

4.1009.4.5 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count ( ) const [inline], [noexcept]

Returns the number of buckets of the unordered_set.

Definition at line 559 of file unordered_set.h.

4.1009.4.6 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

Definition at line 270 of file unordered_set.h.

4.1009.4.7 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 601 of file unordered_set.h.

4.1009.4.8 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

Definition at line 278 of file unordered_set.h.

4.1009.4.9 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 621 of file unordered_set.h.

4.1009.4.10 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::clear ( ) [inline], [noexcept]

Erases all elements in an unordered_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 471 of file unordered_set.h.

4.1009.4.11 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (const key_type & __x) const [inline]

Finds the number of elements.

Parameters

__x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multisets; for unordered_set the result will either be 0 (not present) or 1 (present).

Definition at line 534 of file unordered_set.h.

4.1009.4.12 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace (_Args &&... __args) [inline]

Attempts to build and insert an element into the unordered_set.

Parameters

__args	Arguments used to generate an element.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.

Insertion requires amortized constant time.

Definition at line 300 of file unordered_set.h.

4.1009.4.13 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename... _Args> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace_hint (const_iterator *__pos*, _Args &&... *__args*) [inline]

Attempts to insert an element into the unordered_set.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

> An iterator that points to the element with key equivalent to the one generated from *__args* (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html

Insertion requires amortized constant time.

Definition at line 326 of file unordered_set.h.

4.1009.4.14 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the unordered_set is empty.

Definition at line 222 of file unordered_set.h.

4.1009.4.15 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end ( ) [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

Definition at line 257 of file unordered_set.h.

4.1009.4.16 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

Definition at line 261 of file unordered_set.h.

4.1009.4.17 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (size_type *__n*) [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 613 of file unordered_set.h.

4.1009.4.18 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (size_type *__n*) const [inline]

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read-only local iterator.

Definition at line 617 of file unordered_set.h.

4.1009.4.19 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::pair<iterator, iterator> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (const key_type & *__x*) [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 547 of file unordered_set.h.

4.1009.4.20 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::pair<const_iterator, const_iterator> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (const key_type & *__x*) const [inline]

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 551 of file unordered_set.h.

4.1009.4.21 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (const_iterator *__position*) [inline]

Erases an element from an unordered_set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 421 of file unordered_set.h.

4.1009.4.22 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (iterator *__it*) [inline]

Erases an element from an unordered_set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 426 of file unordered_set.h.

4.1009.4.23 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (const key_type & *__x*) [inline]

Erases elements according to the provided key.

Parameters

__x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_set. For an unordered_set the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 443 of file unordered_set.h.

4.1009.4.24 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (const_iterator *__first,* const_iterator *__last*) [inline]

Erases a [__first,__last) range of elements from an unordered_set.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 461 of file unordered_set.h.

4.1009.4.25 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (const key_type & *__x*) [inline]

Tries to locate an element in an unordered_set.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 516 of file unordered_set.h.

4.1009.4.26 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (const key_type & __x) const [inline]

Tries to locate an element in an unordered_set.

Parameters

__x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 520 of file unordered_set.h.

4.1009.4.27 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator ( ) const [inline], [noexcept]

Returns the allocator object with which the unordered_set was constructed.

Definition at line 215 of file unordered_set.h.

4.1009.4.28 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function ( ) const [inline]

Returns the hash functor object with which the unordered_set was constructed.

Definition at line 492 of file unordered_set.h.

4.1009.4.29 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (const value_type & __x) [inline]

Attempts to insert an element into the unordered_set.

Parameters

__x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.

Insertion requires amortized constant time.

Definition at line 344 of file unordered_set.h.

4.1009.4.30 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (value_type && __x) [inline]

Attempts to insert an element into the unordered_set.

Parameters

__x	Element to be inserted.

Returns

> A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.

Insertion requires amortized constant time.

Definition at line 348 of file unordered_set.h.

4.1009.4.31 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (const_iterator __hint, const value_type & __x) [inline]

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html`

Insertion requires amortized constant.

Definition at line 373 of file unordered_set.h.

4.1009.4.32 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (const_iterator *__hint,* value_type && *__x*) `[inline]`

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt07ch17.html`

Insertion requires amortized constant.

Definition at line 377 of file unordered_set.h.

4.1009.4.33 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> template<typename _InputIterator > void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (_InputIterator *__first,* _InputIterator *__last*) `[inline]`

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

Definition at line 392 of file unordered_set.h.

4.1009.4.34 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >::insert (**initializer_list**< **value_type** > *__l*) [inline]

Attempts to insert a list of elements into the unordered_set.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

Definition at line 403 of file unordered_set.h.

4.1009.4.35 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> **key_equal std::unordered_set**< _Value, _Hash, _Pred, _Alloc >::key_eq ( ) const [inline]

Returns the key comparison object with which the unordered_set was constructed.

Definition at line 498 of file unordered_set.h.

4.1009.4.36 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >::load_factor ( ) const [inline], [noexcept]

Returns the average number of elements per bucket.

Definition at line 629 of file unordered_set.h.

4.1009.4.37 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> **size_type std::unordered_set**< _Value, _Hash, _Pred, _Alloc >::max_bucket_count ( ) const [inline], [noexcept]

Returns the maximum number of buckets of the unordered_set.

Definition at line 564 of file unordered_set.h.

4.1009.4.38 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> float **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >::max_load_factor ( ) const [inline], [noexcept]

Returns a positive number that the unordered_set tries to keep the load factor less than or equal to.

Definition at line 635 of file unordered_set.h.

4.1009.4.39 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void **std::unordered_set**< _Value, _Hash, _Pred, _Alloc >::max_load_factor (float *__z*) [inline]

Change the unordered_set maximum load factor.

Parameters

__z	The new maximum load factor.

Definition at line 643 of file unordered_set.h.

4.1009.4.40 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size ( ) const `[inline]`, `[noexcept]`

Returns the maximum size of the unordered_set.

Definition at line 232 of file unordered_set.h.

4.1009.4.41 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (const unordered_set< _Value, _Hash, _Pred, _Alloc > &) `[default]`

Copy assignment operator.

4.1009.4.42 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (unordered_set< _Value, _Hash, _Pred, _Alloc > &&) `[default]`

Move assignment operator.

4.1009.4.43 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (initializer_list< value_type > __l) `[inline]`

Unordered_set list assignment operator.

Parameters

__l	An initializer_list.

This function fills an unordered_set with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the unordered_set and that the resulting unordered_set's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 206 of file unordered_set.h.

4.1009.4.44 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (size_type *__n*) `[inline]`

May rehash the unordered_set.

Parameters

__n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_set maximum load factor.

Definition at line 654 of file unordered_set.h.

4.1009.4.45 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (size_type *__n*) `[inline]`

Prepare the unordered_set for a specified number of elements.

Parameters

__n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

Definition at line 665 of file unordered_set.h.

4.1009.4.46 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size ( ) const [inline], [noexcept]

Returns the size of the unordered_set.

Definition at line 227 of file unordered_set.h.

4.1009.4.47 template<class _Value, class _Hash = hash<_Value>, class _Pred = std::equal_to<_Value>, class _Alloc = std::allocator<_Value>> void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (unordered_set< _Value, _Hash, _Pred, _Alloc > & __x) [inline]

Swaps data with another unordered_set.

Parameters

__x	An unordered_set of the same element and allocator types.

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

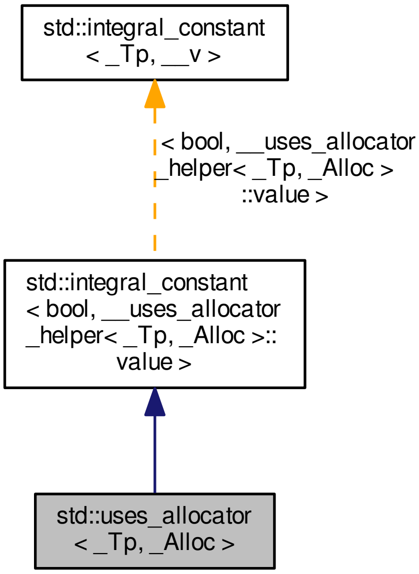
Definition at line 484 of file unordered_set.h.

The documentation for this class was generated from the following file:

- unordered_set.h

4.1010 std::uses_allocator< _Tp, _Alloc > Struct Template Reference

Inheritance diagram for std::uses_allocator< _Tp, _Alloc >:



Public Types

- typedef integral_constant
  < bool, __v > **type**
- typedef bool **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr bool **value**

4.1010.1 Detailed Description

template<typename _Tp, typename _Alloc>struct std::uses_allocator< _Tp, _Alloc >

[allocator.uses.trait]

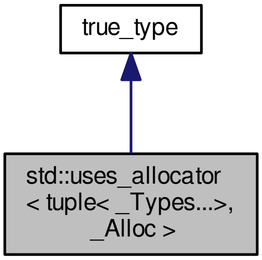
Definition at line 58 of file uses_allocator.h.

The documentation for this struct was generated from the following file:

- uses_allocator.h

4.1011 std::uses_allocator< tuple< _Types...>, _Alloc > Struct Template Reference

Inheritance diagram for std::uses_allocator< tuple< _Types...>, _Alloc >:



Public Types

- typedef integral_constant< _Tp, __v > **type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **operator value_type** ()

Static Public Attributes

- static constexpr _Tp **value**

4.1011.1 Detailed Description

template<typename... _Types, typename _Alloc>struct std::uses_allocator< tuple< _Types...>, _Alloc >

Partial specialization for tuples.

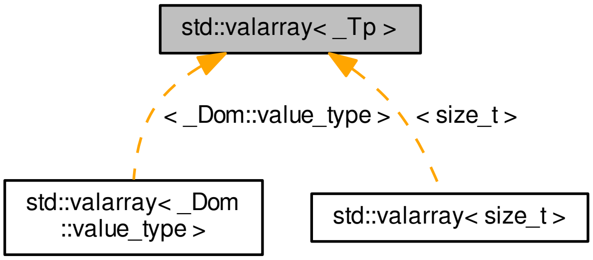
Definition at line 1066 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.1012 std::valarray< _Tp > Class Template Reference

Inheritance diagram for std::valarray< _Tp >:



Public Types

- typedef _Tp **value_type**

Public Member Functions

- valarray ()
- valarray (size_t)
- valarray (const _Tp &, size_t)
- valarray (const _Tp ∗__restrict__, size_t)
- valarray (const valarray &)
- valarray (valarray &&) noexcept
- valarray (const slice_array< _Tp > &)
- valarray (const gslice_array< _Tp > &)
- valarray (const mask_array< _Tp > &)
- valarray (const indirect_array< _Tp > &)
- valarray (initializer_list< _Tp >)
- template<class _Dom >
  valarray (const _Expr< _Dom, _Tp > &__e)
- template<typename _Tp>
  valarray (const _Tp ∗__restrict__ __p, size_t __n)
- _Expr< _ValFunClos< _ValArray,
  _Tp >, _Tp > apply (_Tp func(_Tp)) const
- _Expr< _RefFunClos< _ValArray,
  _Tp >, _Tp > apply (_Tp func(const _Tp &)) const
- valarray< _Tp > cshift (int __n) const
- _Tp max () const
- _Tp min () const
- _UnaryOp< __logical_not >::_Rt operator! () const
- valarray< _Tp > & operator%= (const _Tp &)
- valarray< _Tp > & operator%= (const valarray< _Tp > &)

- template<class _Dom >
  valarray< _Tp > & **operator%=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator&= (const _Tp &)
- valarray< _Tp > & operator&= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator&=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator∗= (const _Tp &)
- valarray< _Tp > & operator∗= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator∗=** (const _Expr< _Dom, _Tp > &)
- _UnaryOp< __unary_plus >::_Rt operator+ () const
- valarray< _Tp > & operator+= (const _Tp &)
- valarray< _Tp > & operator+= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator+=** (const _Expr< _Dom, _Tp > &)
- _UnaryOp< __negate >::_Rt operator- () const
- valarray< _Tp > & operator-= (const _Tp &)
- valarray< _Tp > & operator-= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator-=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator/= (const _Tp &)
- valarray< _Tp > & operator/= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator/=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator<<= (const _Tp &)
- valarray< _Tp > & operator<<= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator<<=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator= (const valarray< _Tp > &__v)
- valarray< _Tp > & operator= (valarray< _Tp > &&__v) noexcept
- valarray< _Tp > & operator= (const _Tp &__t)
- valarray< _Tp > & operator= (const slice_array< _Tp > &__sa)
- valarray< _Tp > & operator= (const gslice_array< _Tp > &__ga)
- valarray< _Tp > & operator= (const mask_array< _Tp > &__ma)
- valarray< _Tp > & operator= (const indirect_array< _Tp > &__ia)
- valarray & operator= (initializer_list< _Tp > __l)
- template<class _Dom >
  valarray< _Tp > & **operator=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator>>= (const _Tp &)
- valarray< _Tp > & operator>>= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator>>=** (const _Expr< _Dom, _Tp > &)
- _Tp & operator[] (size_t __i)
- const _Tp & **operator[]** (size_t) const
- _Expr< _SClos< _ValArray, _Tp >
  , _Tp > operator[] (slice __s) const
- slice_array< _Tp > operator[] (slice __s)
- _Expr< _GClos< _ValArray, _Tp >
  , _Tp > operator[] (const gslice &__s) const
- gslice_array< _Tp > operator[] (const gslice &__s)
- valarray< _Tp > operator[] (const valarray< bool > &__m) const

- mask_array< _Tp > operator[] (const valarray< bool > &__m)
- _Expr< _IClos< _ValArray, _Tp >
  , _Tp > operator[] (const valarray< size_t > &__i) const
- indirect_array< _Tp > operator[] (const valarray< size_t > &__i)
- valarray< _Tp > & operator^= (const _Tp &)
- valarray< _Tp > & operator^= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator^=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator|= (const _Tp &)
- valarray< _Tp > & operator|= (const valarray< _Tp > &)
- template<class _Dom >
  valarray< _Tp > & **operator|=** (const _Expr< _Dom, _Tp > &)
- _UnaryOp< __bitwise_not >::_Rt operator~ () const
- void resize (size_t __size, _Tp __c=_Tp())
- valarray< _Tp > shift (int __n) const
- size_t size () const
- _Tp sum () const
- void swap (valarray< _Tp > &__v) noexcept

Friends

- class **_Array< _Tp >**

4.1012.1 Detailed Description

template<class _Tp>class std::valarray< _Tp >

Smart array designed to support numeric processing.

A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

Template Parameters

_Tp	Type of object in the array.

Definition at line 116 of file valarray.

4.1012.2 Constructor & Destructor Documentation

4.1012.2.1 template<class _Tp> std::valarray< _Tp >::valarray (const _Tp * *__restrict__*, size_t)

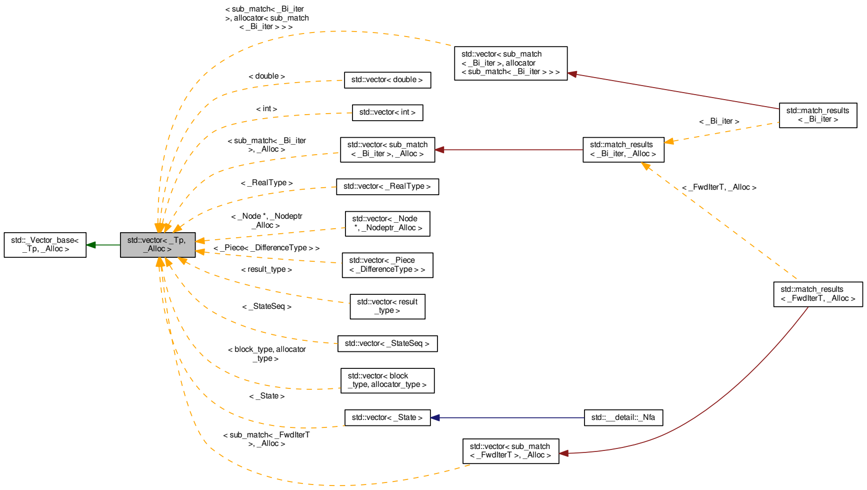
Construct an array initialized to the first *n* elements of *t*.

The documentation for this class was generated from the following file:

- valarray

4.1013 std::vector< _Tp, _Alloc > Class Template Reference

Inheritance diagram for std::vector< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator < const_pointer, vector > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator < pointer, vector > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- vector ()
- vector (const allocator_type &__a)
- vector (size_type __n, const allocator_type &__a=allocator_type())
- vector (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())

- vector (const vector &__x)
- vector (vector &&__x) noexcept
- vector (const vector &__x, const allocator_type &__a)
- vector (vector &&__rv, const allocator_type &__m)
- vector (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- ∼vector () noexcept
- void assign (size_type __n, const value_type &__val)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- reference at (size_type __n)
- const_reference at (size_type __n) const
- reference back ()
- const_reference back () const
- iterator begin () noexcept
- const_iterator begin () const noexcept
- size_type capacity () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- _Tp ∗ data () noexcept
- const _Tp ∗ **data** () const noexcept
- template<typename... _Args>
  iterator emplace (iterator __position, _Args &&...__args)
- template<typename... _Args>
  void **emplace_back** (_Args &&...__args)
- bool empty () const noexcept
- iterator end () noexcept
- const_iterator end () const noexcept
- iterator erase (iterator __position)
- iterator erase (iterator __first, iterator __last)
- reference front ()
- const_reference front () const
- iterator insert (iterator __position, const value_type &__x)
- iterator insert (iterator __position, value_type &&__x)
- void insert (iterator __position, initializer_list< value_type > __l)
- void insert (iterator __position, size_type __n, const value_type &__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void insert (iterator __position, _InputIterator __first, _InputIterator __last)
- size_type max_size () const noexcept
- vector & operator= (const vector &__x)
- vector & operator= (vector &&__x) noexcept(_Alloc_traits
- vector & operator= (initializer_list< value_type > __l)
- reference operator[] (size_type __n)
- const_reference operator[] (size_type __n) const
- void pop_back ()
- void push_back (const value_type &__x)

- void **push_back** (value_type &&__x)
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rend () noexcept
- const_reverse_iterator rend () const noexcept
- void reserve (size_type __n)
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void shrink_to_fit ()
- size_type size () const noexcept
- void swap (vector &__x) noexcept(_Alloc_traits

Protected Member Functions

- pointer **_M_allocate** (size_t __n)
- template<typename _ForwardIterator >
  pointer _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- template<typename _InputIterator >
  void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _Integer >
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator >
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- size_type **_M_check_len** (size_type __n, const char ∗__s) const
- void **_M_deallocate** (pointer __p, size_t __n)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- template<typename... _Args>
  void **_M_emplace_back_aux** (_Args &&...__args)
- void **_M_erase_at_end** (pointer __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- template<typename _Integer >
  void **_M_initialize_dispatch** (_Integer __n, _Integer __value, __true_type)
- template<typename _InputIterator >
  void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename... _Args>
  void **_M_insert_aux** (iterator __position, _Args &&...__args)
- template<typename _Integer >
  void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator >
  void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- void _M_range_check (size_type __n) const
- template<typename _InputIterator >
  void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)

- template<typename _ForwardIterator >
  void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator >
  void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- bool **_M_shrink_to_fit** ()
- allocator_type **get_allocator** () const noexcept

Protected Attributes

- _Vector_impl **_M_impl**

4.1013.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>class std::vector< _Tp, _Alloc >

A standard container which offers fixed time access to individual elements in any order.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements` with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

Definition at line 210 of file stl_vector.h.

4.1013.2 Constructor & Destructor Documentation

4.1013.2.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector ( ) `[inline]`

Default constructor creates no elements.

Definition at line 248 of file stl_vector.h.

4.1013.2.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (const allocator_type & *__a*) `[inline]`, `[explicit]`

Creates a vector with no elements.

Parameters

__a	An allocator object.

Definition at line 256 of file stl_vector.h.

4.1013.2.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (size_type __n, const allocator_type & __a = `allocator_type()`) `[inline]`, `[explicit]`

Creates a vector with default constructed elements.

Parameters

__n	The number of elements to initially create.
__a	An allocator.

This constructor fills the vector with *__n* default constructed elements.

Definition at line 269 of file stl_vector.h.

4.1013.2.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (size_type __n, const value_type & __value, const allocator_type & __a = `allocator_type()`) `[inline]`

Creates a vector with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator.

This constructor fills the vector with *__n* copies of *__value*.

Definition at line 281 of file stl_vector.h.

4.1013.2.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (const vector< _Tp, _Alloc > & __x) `[inline]`

Vector copy constructor.

Parameters

__x	A vector of identical element and allocator types.

The newly-created vector uses a copy of the allocation object used by *__x*. All the elements of *__x* are copied, but any extra memory in *__x* (for fast expansion) will not be copied.

Definition at line 310 of file stl_vector.h.

4.1013.2.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (vector< _Tp, _Alloc > && __x) `[inline]`, `[noexcept]`

Vector move constructor.

Parameters

__x	A vector of identical element and allocator types.

The newly-created vector contains the exact contents of *__x*. The contents of *__x* are a valid, but unspecified vector.

Definition at line 327 of file stl_vector.h.

4.1013.2.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (const vector< _Tp, _Alloc > & *__x*, const allocator_type & *__a*) `[inline]`

Copy constructor with alternative allocator.

Definition at line 331 of file stl_vector.h.

4.1013.2.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (vector< _Tp, _Alloc > && *__rv*, const allocator_type & *__m*) `[inline]`

Move constructor with alternative allocator.

Definition at line 340 of file stl_vector.h.

4.1013.2.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::vector (initializer_list< value_type > *__l*, const allocator_type & *__a* = `allocator_type()`) `[inline]`

Builds a vector from an initializer list.

Parameters

__l	An initializer_list.
__a	An allocator.

Create a vector consisting of copies of the elements in the initializer_list *__l*.

This will call the element type's copy constructor N times (where N is *__l.size()*) and do no memory reallocation.

Definition at line 364 of file stl_vector.h.

4.1013.2.10 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> std::vector< _Tp, _Alloc >::vector (_InputIterator *__first*, _InputIterator *__last*, const allocator_type & *__a* = `allocator_type()`) `[inline]`

Builds a vector from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__a	An allocator.

Create a vector consisting of copies of the elements from [first,last).

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

Definition at line 392 of file stl_vector.h.

4.1013.2.11 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> std::vector< _Tp, _Alloc >::∼vector ( ) `[inline]`, `[noexcept]`

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 414 of file stl_vector.h.

4.1013.3 Member Function Documentation

4.1013.3.1 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _ForwardIterator > pointer std::vector< _Tp, _Alloc >::_M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last) [inline], [protected]

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

Definition at line 1135 of file stl_vector.h.

4.1013.3.2 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::_M_range_check (size_type __n) const [inline], [protected]

Safety check used only from at().

Definition at line 791 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::at().

4.1013.3.3 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::assign (size_type __n, const value_type & __val) [inline]

Assigns a given value to a vector.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a vector with *__n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 479 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::operator=().

4.1013.3.4 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void std::vector< _Tp, _Alloc >::assign (_InputIterator __first, _InputIterator __last) [inline]

Assigns a range to a vector.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a vector with copies of the elements in the range [__first,__last).

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 498 of file stl_vector.h.

4.1013.3.5 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::assign (initializer_list< value_type > __l) [inline]

Assigns an initializer list to a vector.

Parameters

__l	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 524 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::assign().

4.1013.3.6 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::at (size_type *__n*) [inline]

Provides access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

Definition at line 810 of file stl_vector.h.

4.1013.3.7 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc >::at (size_type *__n*) const [inline]

Provides access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

Definition at line 828 of file stl_vector.h.

4.1013.3.8 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::back ( ) [inline]

Returns a read/write reference to the data at the last element of the vector.

Definition at line 855 of file stl_vector.h.

Referenced by std::piecewise_constant_distribution< _RealType >::max(), and std::piecewise_linear_distribution< _RealType >::max().

4.1013.3.9 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc >::back ( ) const [inline]

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 863 of file stl_vector.h.

4.1013.3.10 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::begin ( ) [inline], [noexcept]

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 538 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::crend(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::front(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge_exact_splitting(), std::operator<(), std::vector< _Tp, _Alloc >::operator=(), std::operator==(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rend(), and std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector().

4.1013.3.11 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector< _Tp, _Alloc >::begin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 547 of file stl_vector.h.

4.1013.3.12 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector< _Tp, _Alloc >::capacity ( ) const [inline], [noexcept]

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 725 of file stl_vector.h.

4.1013.3.13 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector< _Tp, _Alloc >::cbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 611 of file stl_vector.h.

4.1013.3.14 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator std::vector< _Tp, _Alloc >::cend ( ) const [inline], [noexcept]

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 620 of file stl_vector.h.

4.1013.3.15 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::clear ( ) [inline], [noexcept]

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1125 of file stl_vector.h.

4.1013.3.16 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 629 of file stl_vector.h.

4.1013.3.17 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector< _Tp, _Alloc >::crend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 638 of file stl_vector.h.

4.1013.3.18 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> _Tp* std::vector< _Tp, _Alloc >::data ( ) [inline], [noexcept]

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

Definition at line 878 of file stl_vector.h.

4.1013.3.19 template<typename _Tp , typename _Alloc > template<typename... _Args> vector< _Tp, _Alloc >::iterator vector::emplace (iterator *__position*, _Args &&... *__args*)

Inserts an object in vector before specified iterator.

Parameters

__position	An iterator into the vector.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 292 of file vector.tcc.

References std::begin(), and std::end().

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert().

4.1013.3.20 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> bool std::vector< _Tp, _Alloc >::empty ( ) const [inline], [noexcept]

Returns true if the vector is empty. (Thus begin() would equal end().)

Definition at line 734 of file stl_vector.h.

Referenced by std::piecewise_constant_distribution< _RealType >::densities(), std::piecewise_linear_distribution< _RealType >::densities(), std::piecewise_constant_distribution< _RealType >::intervals(), std::piecewise_linear_distribution< _RealType >::intervals(), std::discrete_distribution< _IntType >::max(), std::piecewise_constant_distribution< _RealType >::max(), std::piecewise_linear_distribution< _RealType >::max(), std::piecewise_constant_distribution< _RealType >::min(), std::piecewise_linear_distribution< _RealType >::min(), and std::discrete_distribution< _IntType >::probabilities().

4.1013.3.21 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator **std::vector**< _Tp, _Alloc >::end ( ) `[inline]`, `[noexcept]`

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 556 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::back(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::crbegin(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge_exact_splitting(), std::operator<(), std::vector< _Tp, _Alloc >::operator=(), std::operator==(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::push_back(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rbegin(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize(), and std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector().

4.1013.3.22 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_iterator **std::vector**< _Tp, _Alloc >::end ( ) const `[inline]`, `[noexcept]`

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 565 of file stl_vector.h.

4.1013.3.23 template<typename _Tp , typename _Alloc > **vector**< _Tp, _Alloc >::iterator vector::erase (iterator *__position*)

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 135 of file vector.tcc.

References std::end().

4.1013.3.24 template<typename _Tp , typename _Alloc > **vector**< _Tp, _Alloc >::iterator vector::erase (iterator *__first*, iterator *__last*)

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *__last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 147 of file vector.tcc.

References std::end().

4.1013.3.25 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::front ( ) `[inline]`

Returns a read/write reference to the data at the first element of the vector.

Definition at line 839 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::data(), std::piecewise_constant_distribution< _RealType >::min(), and std::piecewise_linear_distribution< _RealType >::min().

4.1013.3.26 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc >::front ( ) const `[inline]`

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 847 of file stl_vector.h.

4.1013.3.27 template<typename _Tp , typename _Alloc > vector< _Tp, _Alloc >::iterator vector::insert (iterator *__position,* const value_type & *__x*)

Inserts given value into vector before specified iterator.

Parameters

__position	An iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 108 of file vector.tcc.

References std::begin(), and std::end().

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize().

4.1013.3.28 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> iterator std::vector< _Tp, _Alloc >::insert (iterator *__position*, value_type && *__x*) [inline]

Inserts given rvalue into vector before specified iterator.

Parameters

__position	An iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 988 of file stl_vector.h.

4.1013.3.29 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::insert (iterator *__position*, initializer_list< value_type > *__l*) [inline]

Inserts an initializer_list into the vector.

Parameters

__position	An iterator into the vector.
__l	An initializer_list.

This function will insert copies of the data in the initializer_list *l* into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1005 of file stl_vector.h.

Referenced by std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert().

4.1013.3.30 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::insert (iterator *__position*, size_type *__n*, const value_type & *__x*) [inline]

Inserts a number of copies of given data into the vector.

Parameters

__position	An iterator into the vector.
__n	Number of elements to be inserted.
__x	Data to be inserted.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1023 of file stl_vector.h.

4.1013.3.31 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>> void **std::vector**< _Tp, _Alloc >::insert (iterator *__position,* _InputIterator *__first,* _InputIterator *__last*) `[inline]`

Inserts a range into the vector.

Parameters

__position	An iterator into the vector.
__first	An input iterator.
__last	An input iterator.

This function will insert copies of the data in the range [__first,__last) into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1044 of file stl_vector.h.

4.1013.3.32 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type **std::vector**< _Tp, _Alloc >::max_size ( ) const `[inline]`, `[noexcept]`

Returns the size() of the largest possible vector.

Definition at line 650 of file stl_vector.h.

4.1013.3.33 template<typename _Tp , typename _Alloc > **vector**< _Tp, _Alloc > & vector::operator= (const **vector**< _Tp, _Alloc > & *__x*)

Vector assignment operator.

Parameters

__x	A vector of identical element and allocator types.

All the elements of *__x* are copied, but any extra memory in *__x* (for fast expansion) will not be copied. Unlike the copy constructor, the allocator object is not copied.

Definition at line 161 of file vector.tcc.

References std::_Destroy(), std::begin(), std::vector< _Tp, _Alloc >::begin(), std::end(), std::vector< _Tp, _Alloc >::end(), std::vector< _Tp, _Alloc >::size(), and std::size().

4.1013.3.34 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> **vector&** **std::vector**< _Tp, _Alloc >::operator= (**vector**< _Tp, _Alloc > && *__x*) `[inline]`, `[noexcept]`

Vector move assignment operator.

Parameters

__x	A vector of identical element and allocator types.

The contents of *__x* are moved into this vector (without copying, if the allocators permit it). *__x* is a valid, but unspecified vector.

Definition at line 439 of file stl_vector.h.

4.1013.3.35 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> vector& std::vector< _Tp, _Alloc >::operator= (initializer_list< value_type > __l) [inline]

Vector list assignment operator.

Parameters

__l	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned. Old data may be lost.

Definition at line 461 of file stl_vector.h.

4.1013.3.36 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reference std::vector< _Tp, _Alloc >::operator[] (size_type __n) [inline]

Subscript access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 770 of file stl_vector.h.

4.1013.3.37 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reference std::vector< _Tp, _Alloc >::operator[] (size_type __n) const [inline]

Subscript access to the data contained in the vector.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

Definition at line 785 of file stl_vector.h.

4.1013.3.38 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::pop_back ( ) [inline]

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

Definition at line 937 of file stl_vector.h.

4.1013.3.39 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::push_back (const value_type & __x) [inline]

Add data to the end of the vector.

Parameters

__x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 901 of file stl_vector.h.

Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

4.1013.3.40 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector< _Tp, _Alloc >::rbegin ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 574 of file stl_vector.h.

4.1013.3.41 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 583 of file stl_vector.h.

4.1013.3.42 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> reverse_iterator std::vector< _Tp, _Alloc >::rend ( ) [inline], [noexcept]

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 592 of file stl_vector.h.

4.1013.3.43 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> const_reverse_iterator std::vector< _Tp, _Alloc >::rend ( ) const [inline], [noexcept]

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 601 of file stl_vector.h.

4.1013.3.44 template<typename _Tp , typename _Alloc > void vector::reserve (size_type __n)

Attempt to preallocate enough memory for specified number of elements.

Parameters

__n	Number of elements required.

Exceptions

std::length_error	If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than max_size(), length_error is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 66 of file vector.tcc.

References std::_Destroy(), and std::size().

4.1013.3.45 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::resize (size_type *__new_size*) `[inline]`

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 664 of file stl_vector.h.

Referenced by __gnu_parallel::__shrink_and_double(), __gnu_parallel::multiway_merge_exact_splitting(), and __gnu_parallel::parallel_sort_mwms().

4.1013.3.46 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::resize (size_type *__new_size*, const value_type & *__x*) `[inline]`

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.
__x	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 684 of file stl_vector.h.

4.1013.3.47 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::shrink_to_fit ( ) `[inline]`

A non-binding request to reduce capacity() to size().

Definition at line 716 of file stl_vector.h.

4.1013.3.48 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> size_type std::vector< _Tp, _Alloc >::size ( ) const `[inline]`, `[noexcept]`

Returns the number of elements in the vector.

Definition at line 645 of file stl_vector.h.

Referenced by __gnu_parallel::__shrink(), __gnu_parallel::__shrink_and_double(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::_M_range_check(), __gnu_parallel::list_partition(), std::discrete_distribution< _IntType >::max(), std::vector< _Tp, _Alloc >::operator=(), std::operator==(), and std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize().

4.1013.3.49 template<typename _Tp, typename _Alloc = std::allocator<_Tp>> void std::vector< _Tp, _Alloc >::swap (vector< _Tp, _Alloc > & *__x*) [inline], [noexcept]

Swaps data with another vector.

Parameters

__x	A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(v1,v2) will feed to this function.

Definition at line 1108 of file stl_vector.h.

Referenced by std::swap().

The documentation for this class was generated from the following files:

- stl_vector.h
- vector.tcc

4.1014 std::vector< bool, _Alloc > Class Template Reference

Inherits std::_Bvector_base< _Alloc >.

Public Types

- typedef _Alloc **allocator_type**
- typedef _Bit_const_iterator **const_iterator**
- typedef const bool ∗ **const_pointer**
- typedef bool **const_reference**
- typedef std::reverse_iterator < const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Bit_iterator **iterator**
- typedef _Bit_reference ∗ **pointer**
- typedef _Bit_reference **reference**
- typedef std::reverse_iterator < iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef bool **value_type**

Public Member Functions

- **vector** (const allocator_type &__a)
- **vector** (size_type __n, const allocator_type &__a=allocator_type())
- **vector** (size_type __n, const bool &__value, const allocator_type &__a=allocator_type())
- **vector** (const vector &__x)

- **vector** (vector &&__x) noexcept
- **vector** (initializer_list< bool > __l, const allocator_type &__a=allocator_type())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- void **assign** (size_type __n, const bool &__x)
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (initializer_list< bool > __l)
- reference **at** (size_type __n)
- const_reference **at** (size_type __n) const
- reference **back** ()
- const_reference **back** () const
- iterator **begin** () noexcept
- const_iterator **begin** () const noexcept
- size_type **capacity** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- void **data** () noexcept
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const_iterator **end** () const noexcept
- iterator **erase** (iterator __position)
- iterator **erase** (iterator __first, iterator __last)
- void **flip** () noexcept
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (iterator __position, const bool &__x=bool())
- template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
  void **insert** (iterator __position, _InputIterator __first, _InputIterator __last)
- void **insert** (iterator __position, size_type __n, const bool &__x)
- void **insert** (iterator __p, initializer_list< bool > __l)
- size_type **max_size** () const noexcept
- vector & **operator=** (const vector &__x)
- vector & **operator=** (vector &&__x)
- vector & **operator=** (initializer_list< bool > __l)
- reference **operator[]** (size_type __n)
- const_reference **operator[]** (size_type __n) const
- void **pop_back** ()
- void **push_back** (bool __x)
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** () noexcept
- const_reverse_iterator **rend** () const noexcept
- void **reserve** (size_type __n)
- void **resize** (size_type __new_size, bool __x=bool())
- void **shrink_to_fit** ()
- size_type **size** () const noexcept
- void **swap** (vector &__x)

Static Public Member Functions

- static void **swap** (reference __x, reference __y) noexcept

Protected Types

- typedef _Alloc::template
  rebind< _Bit_type >::other **_Bit_alloc_type**

Protected Member Functions

- _Bit_type ∗ **_M_allocate** (size_t __n)
- template<typename _InputIterator >
  void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _Integer >
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- template<class _InputIterator >
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- size_type **_M_check_len** (size_type __n, const char ∗__s) const
- iterator **_M_copy_aligned** (const_iterator __first, const_iterator __last, iterator __result)
- void **_M_deallocate** ()
- void **_M_erase_at_end** (iterator __pos)
- void **_M_fill_assign** (size_t __n, bool __x)
- void **_M_fill_insert** (iterator __position, size_type __n, bool __x)
- _Bit_alloc_type & **_M_get_Bit_allocator** () noexcept
- const _Bit_alloc_type & **_M_get_Bit_allocator** () const noexcept
- void **_M_initialize** (size_type __n)
- template<typename _Integer >
  void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
  void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _InputIterator >
  void **_M_initialize_range** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_initialize_range** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_insert_aux** (iterator __position, bool __x)
- template<typename _Integer >
  void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __x, __true_type)
- template<typename _InputIterator >
  void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- template<typename _InputIterator >
  void **_M_insert_range** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator >
  void **_M_insert_range** (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- void **_M_range_check** (size_type __n) const
- void **_M_reallocate** (size_type __n)
- bool **_M_shrink_to_fit** ()

Static Protected Member Functions

- static size_t **_S_nword** (size_t __n)

Protected Attributes

- _Bvector_impl **_M_impl**

Friends

- template<typename >
  class **hash**

4.1014.1 Detailed Description

template<typename _Alloc>class std::vector< bool, _Alloc >

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Template Parameters

_Alloc	Allocator type.

Note that vector<bool> does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to bool. See DR96 for details.

See Also

vector for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

Definition at line 518 of file stl_bvector.h.

The documentation for this class was generated from the following files:

- stl_bvector.h
- vector.tcc

4.1015 std::weak_ptr< _Tp > Class Template Reference

Inherits std::__weak_ptr< _Tp, _Lp >.

Public Types

- typedef _Tp **element_type**

Public Member Functions

- template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1*, _Tp*>::value>::type>
  weak_ptr (const weak_ptr< _Tp1 > &__r) noexcept

- template<typename _Tp1 , typename = typename std::enable_if<std::is_convertible<_Tp1∗, _Tp∗>::value>::type>
  weak_ptr (const shared_ptr< _Tp1 > &__r) noexcept
- bool **expired** () const noexcept
- shared_ptr< _Tp > **lock** () const noexcept
- template<typename _Tp1 >
  weak_ptr & **operator=** (const weak_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp1 >
  weak_ptr & **operator=** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp1 >
  bool **owner_before** (const __shared_ptr< _Tp1, _Lp > &__rhs) const
- template<typename _Tp1 >
  bool **owner_before** (const __weak_ptr< _Tp1, _Lp > &__rhs) const
- void **reset** () noexcept
- void **swap** (__weak_ptr &__s) noexcept
- long **use_count** () const noexcept

4.1015.1 Detailed Description

template<typename _Tp>class std::weak_ptr< _Tp >

A smart pointer with weak semantics.

With forwarding constructors and assignment operators.

Definition at line 461 of file shared_ptr.h.

The documentation for this class was generated from the following file:

- shared_ptr.h

4.1016 std::weibull_distribution< _RealType > Class Template Reference

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **weibull_distribution** (_RealType __a=_RealType(1), _RealType __b=_RealType(1))
- **weibull_distribution** (const param_type &__p)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator , typename _UniformRandomNumberGenerator >
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator >
  void **__generate** (result_type ∗__f, result_type ∗__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- _RealType a () const
- _RealType b () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator >
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator >
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const weibull_distribution &__d1, const weibull_distribution &__d2)

4.1016.1 Detailed Description

template<typename _RealType = double>class std::weibull_distribution< _RealType >

A weibull_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha,\beta) = \frac{\alpha}{\beta} (\frac{x}{\beta})^{\alpha-1} \exp{(-(\frac{x}{\beta})^{\alpha})}$$

Definition at line 4847 of file random.h.

4.1016.2 Member Typedef Documentation

4.1016.2.1 template<typename _RealType = double> typedef _RealType std::weibull_distribution< _RealType >::result_type

The type of the range of the distribution.

Definition at line 4850 of file random.h.

4.1016.3 Member Function Documentation

4.1016.3.1 template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::a ( ) const [inline]

Return the a parameter of the distribution.

Definition at line 4905 of file random.h.

4.1016.3.2 template<typename _RealType = double> _RealType std::weibull_distribution< _RealType >::b ( ) const [inline]

Return the b parameter of the distribution.

Definition at line 4912 of file random.h.

4.1016.3.3 template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::max ( ) const [inline]

Returns the least upper bound value of the distribution.

Definition at line 4941 of file random.h.

References std::numeric_limits< _Tp >::max().

4.1016.3.4 template<typename _RealType = double> result_type std::weibull_distribution< _RealType >::min ( ) const [inline]

Returns the greatest lower bound value of the distribution.

Definition at line 4934 of file random.h.

4.1016.3.5 template<typename _RealType = double> template<typename _UniformRandomNumberGenerator > result_type std::weibull_distribution< _RealType >::operator() (_UniformRandomNumberGenerator & *__urng*) [inline]

Generating functions.

Definition at line 4949 of file random.h.

References std::weibull_distribution< _RealType >::operator()().

Referenced by std::weibull_distribution< _RealType >::operator()().

4.1016.3.6 template<typename _RealType = double> param_type std::weibull_distribution< _RealType >::param ( ) const [inline]

Returns the parameter set of the distribution.

Definition at line 4919 of file random.h.

Referenced by std::operator>>().

4.1016.3.7 template<typename _RealType = double> void std::weibull_distribution< _RealType >::param (const param_type & *__param*) [inline]

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

Definition at line 4927 of file random.h.

4.1016.3.8 template<typename _RealType = double> void std::weibull_distribution< _RealType >::reset ( ) [inline]

Resets the distribution state.

Definition at line 4898 of file random.h.

4.1016.4 Friends And Related Function Documentation

4.1016.4.1 template<typename _RealType = double> bool operator== (const weibull_distribution< _RealType > & *__d1*, const weibull_distribution< _RealType > & *__d2*) [friend]

Return true if two Weibull distributions have the same parameters.

Definition at line 4984 of file random.h.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

4.1017 std::weibull_distribution< _RealType >::param_type Struct Reference

Public Types

- typedef weibull_distribution
  < _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a=_RealType(1), _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

4.1017.1 Detailed Description

template<typename _RealType = double>struct std::weibull_distribution< _RealType >::param_type

Parameter type.

Definition at line 4856 of file random.h.

The documentation for this struct was generated from the following file:

- random.h

4.1018 tree_metadata_helper< Node_Update, _BTp > Struct Template Reference

4.1018.1 Detailed Description

template<typename Node_Update, bool _BTp>struct tree_metadata_helper< Node_Update, _BTp >

Tree metadata helper.

Definition at line 58 of file tree_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

4.1019 tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference

4.1019.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>struct tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >

Tree traits class, primary template.

Definition at line 70 of file branch_policy/traits.hpp.

The documentation for this struct was generated from the following file:

- branch_policy/traits.hpp

4.1020 trie_metadata_helper< Node_Update, _BTp > Struct Template Reference

4.1020.1 Detailed Description

template<typename Node_Update, bool _BTp>struct trie_metadata_helper< Node_Update, _BTp >

Trie metadata helper.

Definition at line 58 of file trie_policy/node_metadata_selector.hpp.

The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

4.1021 trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference

4.1021.1 Detailed Description

template<typename Key, typename Data, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>struct trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >

Trie traits class, primary template.

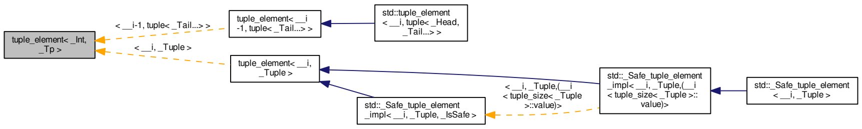
Definition at line 83 of file branch_policy/traits.hpp.

The documentation for this struct was generated from the following file:

- branch_policy/traits.hpp

4.1022 tuple_element< _Int, _Tp > Class Template Reference

Inheritance diagram for tuple_element< _Int, _Tp >:



4.1022.1 Detailed Description

template< std::size_t _Int, typename _Tp > class tuple_element< _Int, _Tp >

tuple_element

Definition at line 311 of file array.

The documentation for this class was generated from the following file:

- array

4.1023 tuple_element< __i, _Tp > Struct Template Reference

4.1023.1 Detailed Description

template< std::size_t __i, typename _Tp > struct tuple_element< __i, _Tp >

Gives the type of the ith element of a given tuple type.

Definition at line 673 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.1024 tuple_size< _Tp > Class Template Reference

4.1024.1 Detailed Description

template< typename _Tp > class tuple_size< _Tp >

tuple_size

Definition at line 303 of file array.

The documentation for this class was generated from the following file:

- array

4.1025 tuple_size< _Tp > Struct Template Reference

4.1025.1 Detailed Description

template<typename _Tp>struct tuple_size< _Tp >

Finds the size of a given tuple type.

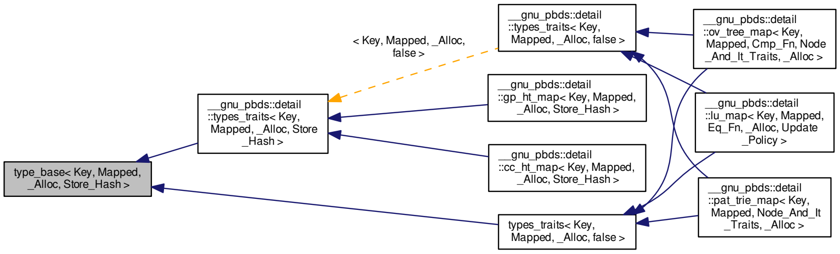
Definition at line 715 of file tuple.

The documentation for this struct was generated from the following file:

- tuple

4.1026 type_base< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

Inheritance diagram for type_base< Key, Mapped, _Alloc, Store_Hash >:



4.1026.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>struct type_base< Key, Mapped, _Alloc, Store_Hash >

Primary template.

Definition at line 107 of file types_traits.hpp.

The documentation for this struct was generated from the following file:

- types_traits.hpp

4.1027 uses_allocator< typename, typename > Struct Template Reference

4.1027.1 Detailed Description

template<typename, typename>struct uses_allocator< typename, typename >

Declare uses_allocator so it can be specialized in <queue> etc.

Definition at line 155 of file allocator.h.

The documentation for this struct was generated from the following file:

- allocator.h

5 File Documentation

5.1 algo.h File Reference

Classes

- struct std::__parallel::_CRandNumber< _MustBeInt >

  Functor wrapper for std::rand().

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _RAIter >
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _FIterator , typename _IteratorTag >
  _FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _FIterator , typename _BinaryPredicate , typename _IteratorTag >
  _FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)
- template<typename _RAIter , typename _BinaryPredicate >
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)
- template<typename _RAIter , typename _Predicate >
  iterator_traits< _RAIter >
  ::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _Predicate , typename _IteratorTag >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::__count_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter , typename _Tp >
  iterator_traits< _RAIter >
  ::difference_type **std::__parallel::__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _Tp , typename _IteratorTag >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::__count_switch** (_IIter __begin, _IIter __end, const _Tp &__value, _IteratorTag)
- template<typename _IIter , typename _FIterator , typename _IteratorTag1 , typename _IteratorTag2 >
  _IIter **std::__parallel::__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)

- template<typename _RAIter , typename _FIterator , typename _BinaryPredicate , typename _IteratorTag >
  _RAIter **std::__parallel::__find_first_of_switch** (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)
- template<typename _IIter , typename _FIterator , typename _BinaryPredicate , typename _IteratorTag1 , typename _IteratorTag2 >
  _IIter **std::__parallel::__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _Predicate , typename _IteratorTag >
  _IIter **std::__parallel::__find_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter , typename _Predicate >
  _RAIter **std::__parallel::__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter , typename _Tp , typename _IteratorTag >
  _IIter **std::__parallel::__find_switch** (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)
- template<typename _RAIter , typename _Tp >
  _RAIter **std::__parallel::__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter , typename _Function , typename _IteratorTag >
  _Function **std::__parallel::__for_each_switch** (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)
- template<typename _RAIter , typename _Function >
  _Function **std::__parallel::__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _OutputIterator , typename _Size , typename _Generator , typename _IteratorTag >
  _OutputIterator **std::__parallel::__generate_n_switch** (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)
- template<typename _RAIter , typename _Size , typename _Generator >
  _RAIter **std::__parallel::__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIterator , typename _Generator , typename _IteratorTag >
  void **std::__parallel::__generate_switch** (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)
- template<typename _RAIter , typename _Generator >
  void **std::__parallel::__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIterator , typename _Compare , typename _IteratorTag >
  _FIterator **std::__parallel::__max_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter , typename _Compare >
  _RAIter **std::__parallel::__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **std::__parallel::__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare >
  _OutputIterator **std::__parallel::__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator , typename _Compare , typename _IteratorTag >
  _FIterator **std::__parallel::__min_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter , typename _Compare >
  _RAIter **std::__parallel::__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

- template<typename _FIterator , typename _Predicate , typename _IteratorTag >
  _FIterator **std::__parallel::__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter , typename _Predicate >
  _RAIter **std::__parallel::__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIterator , typename _Predicate , typename _Tp , typename _IteratorTag >
  void **std::__parallel::__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter , typename _Predicate , typename _Tp >
  void **std::__parallel::__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIterator , typename _Tp , typename _IteratorTag >
  void **std::__parallel::__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter , typename _Tp >
  void **std::__parallel::__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _RAIter **std::__parallel::__search_n_switch** (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)
- template<typename _FIterator , typename _Integer , typename _Tp , typename _BinaryPredicate , typename _IteratorTag >
  _FIterator **std::__parallel::__search_n_switch** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)
- template<typename _RAIter1 , typename _RAIter2 >
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1 , typename _FIterator2 , typename _IteratorTag1 , typename _IteratorTag2 >
  _FIterator1 **std::__parallel::__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _BinaryPredicate >
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1 , typename _FIterator2 , typename _BinaryPredicate , typename _IteratorTag1 , typename _IteratorTag2 >
  _FIterator1 **std::__parallel::__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **std::__parallel::__set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **std::__parallel::__set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **std::__parallel::__set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OutputIterator , typename _IteratorTag1 , typename _IteratorTag2 , typename _IteratorTag3 >
  _OutputIterator **std::__parallel::__set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation >
  _RAIter2 **std::__parallel::__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _UnaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >
  _RAIter2 **std::__parallel::__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BinaryOperation >
  _RAIter3 **std::__parallel::__transform2_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >
  _OutputIterator **std::__parallel::__transform2_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)
- template<typename _IIter , typename _OutputIterator , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >
  _OutputIterator **std::__parallel::__unique_copy_switch** (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter , typename RandomAccessOutputIterator , typename _Predicate >
  RandomAccessOutputIterator **std::__parallel::__unique_copy_switch** (_RAIter __begin, _RAIter __last, RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator >
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _BinaryPredicate >
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator , typename _BinaryPredicate >
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)

- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter , typename _Tp >
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter , typename _FIterator >
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIterator , typename _BinaryPredicate >
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIterator , typename _BinaryPredicate >
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)
- template<typename _IIter , typename _FIterator >
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2)
- template<typename _IIter , typename _Predicate >
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Predicate >
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter , typename _Function >
  _Function **std::__parallel::for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _Iterator , typename _Function >
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _Iterator , typename _Function >
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f)
- template<typename _FIterator , typename _Generator >
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Generator >
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Generator >
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen)

- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen)
- template<typename _FIterator >
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator >
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator , typename _Compare >
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare >
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Compare >
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)
- template<typename _FIterator >
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator >
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator >
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator , typename _Compare >
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Compare >
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)

- template<typename _RAIter , typename _Compare >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _FIterator , typename _Predicate >
  _FIterator **std::__parallel::partition** (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Predicate >
  _FIterator **std::__parallel::partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)
- template<typename _RAIter >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _FIterator , typename _Tp >
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Tp >
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Tp >
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _FIterator , typename _Predicate , typename _Tp >
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Predicate , typename _Tp >
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator , typename _Predicate , typename _Tp >
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)
- template<typename _FIterator1 , typename _FIterator2 >
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)

- template<typename _FIterator1 , typename _FIterator2 >
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)
- template<typename _FIterator1 , typename _FIterator2 , typename _BinaryPredicate >
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator1 , typename _FIterator2 , typename _BinaryPredicate >
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)
- template<typename _FIterator , typename _Integer , typename _Tp >
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator , typename _Integer , typename _Tp >
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)
- template<typename _FIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)

- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator >
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare , typename _Parallelism >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare , typename _Parallelism >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end)

- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out)
- template<typename _IIter , typename _OutputIterator , typename _Predicate >
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred)

5.1.1 Detailed Description

Parallel STL function calls corresponding to the stl_algo.h header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file algo.h.

5.2 algobase.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template< typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >
  bool **std::__parallel::__lexicographical_compare_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template< typename _RAIter1 , typename _RAIter2 , typename _Predicate >
  bool **std::__parallel::__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IteratorTag1 , typename _IteratorTag2 >
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template< typename _RAIter1 , typename _RAIter2 , typename _Predicate >
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template< typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template< typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template< typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template< typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template< typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template< typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template< typename _IIter1 , typename _IIter2 , typename _Predicate >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)

5.2.1 Detailed Description

Parallel STL function calls corresponding to the stl_algobase.h header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file algobase.h.

5.3 algorithm File Reference

Macros

- #define **_GLIBCXX_ALGORITHM**

5.3.1 Detailed Description

This is a Standard C++ Library header.

Definition in file algorithm.

5.4 algorithm File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_ALGORITHM**

Functions

- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  pair< _InputIterator,
  _OutputIterator > **__gnu_cxx::__copy_n** (_InputIterator __first, _Size __count, _OutputIterator __result, input_iterator_tag)
- template<typename _RAIterator , typename _Size , typename _OutputIterator >
  pair< _RAIterator,
  _OutputIterator > **__gnu_cxx::__copy_n** (_RAIterator __first, _Size __count, _OutputIterator __result, random_access_iterator_tag)
- template<typename _InputIterator1 , typename _InputIterator2 >
  int **__gnu_cxx::__lexicographical_compare_3way** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- int **__gnu_cxx::__lexicographical_compare_3way** (const unsigned char ∗__first1, const unsigned char ∗__last1, const unsigned char ∗__first2, const unsigned char ∗__last2)
- int **__gnu_cxx::__lexicographical_compare_3way** (const char ∗__first1, const char ∗__last1, const char ∗__first2, const char ∗__last2)
- template<typename _Tp >
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp , typename _Compare >
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)

- template<typename _InputIterator , typename _RandomAccessIterator , typename _Distance >
  _RandomAccessIterator **__gnu_cxx::__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator , typename _Distance >
  _RandomAccessIterator **__gnu_cxx::__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  pair< _InputIterator,
  _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)
- template<typename _InputIterator , typename _Tp , typename _Size >
  void **__gnu_cxx::count** (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- template<typename _InputIterator , typename _Predicate , typename _Size >
  void **__gnu_cxx::count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- template<typename _InputIterator1 , typename _InputIterator2 >
  int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator , typename _RandomAccessIterator >
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator >
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator , typename _OutputIterator , typename _Distance >
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename _RandomNumberGenerator >
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)

5.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file ext/algorithm.

5.5 algorithm File Reference

Macros

- #define **_PARALLEL_ALGORITHM**

5.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file parallel/algorithm.

5.6 algorithmfwd.h File Reference

Namespaces

- namespace std

Functions

- template<typename _FIter >
  _FIter **std::adjacent_find** (_FIter, _FIter)
- template<typename _FIter , typename _BinaryPredicate >
  _FIter **std::adjacent_find** (_FIter, _FIter, _BinaryPredicate)
- template<typename _IIter , typename _Predicate >
  bool **std::all_of** (_IIter, _IIter, _Predicate)
- template<typename _IIter , typename _Predicate >
  bool **std::any_of** (_IIter, _IIter, _Predicate)
- template<typename _FIter , typename _Tp >
  bool **std::binary_search** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  bool **std::binary_search** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _IIter , typename _OIter >
  _OIter **std::copy** (_IIter, _IIter, _OIter)
- template<typename _BIter1 , typename _BIter2 >
  _BIter2 **std::copy_backward** (_BIter1, _BIter1, _BIter2)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **std::copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter , typename _Size , typename _OIter >
  _OIter **std::copy_n** (_IIter, _Size, _OIter)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::count** (_IIter, _IIter, const _Tp &)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::count_if** (_IIter, _IIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::equal** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
  bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _FIter , typename _Tp >
  pair< _FIter, _FIter > **std::equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  pair< _FIter, _FIter > **std::equal_range** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _FIter , typename _Tp >
  void **std::fill** (_FIter, _FIter, const _Tp &)
- template<typename _OIter , typename _Size , typename _Tp >
  _OIter **std::fill_n** (_OIter, _Size, const _Tp &)
- template<typename _IIter , typename _Tp >
  _IIter **std::find** (_IIter, _IIter, const _Tp &)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **std::find_end** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
  _FIter1 **std::find_end** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **std::find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
  _FIter1 **std::find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _IIter , typename _Predicate >
  _IIter **std::find_if** (_IIter, _IIter, _Predicate)

- template<typename _IIter , typename _Predicate >
  _IIter **std::find_if_not** (_IIter, _IIter, _Predicate)
- template<typename _IIter , typename _Funct >
  _Funct **std::for_each** (_IIter, _IIter, _Funct)
- template<typename _FIter , typename _Generator >
  void **std::generate** (_FIter, _FIter, _Generator)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **std::generate_n** (_OIter, _Size, _Generator)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::includes** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _Compare >
  bool **std::includes** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _BIter >
  void **std::inplace_merge** (_BIter, _BIter, _BIter)
- template<typename _BIter , typename _Compare >
  void **std::inplace_merge** (_BIter, _BIter, _BIter, _Compare)
- template<typename _RAIter >
  bool **std::is_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  bool **std::is_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter >
  _RAIter **std::is_heap_until** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  _RAIter **std::is_heap_until** (_RAIter, _RAIter, _Compare)
- template<typename _IIter , typename _Predicate >
  bool **std::is_partitioned** (_IIter, _IIter, _Predicate)
- template<typename _FIter1 , typename _FIter2 >
  bool **std::is_permutation** (_FIter1, _FIter1, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
  bool **std::is_permutation** (_FIter1, _FIter1, _FIter2, _BinaryPredicate)
- template<typename _FIter >
  bool **std::is_sorted** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  bool **std::is_sorted** (_FIter, _FIter, _Compare)
- template<typename _FIter >
  _FIter **std::is_sorted_until** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  _FIter **std::is_sorted_until** (_FIter, _FIter, _Compare)
- template<typename _FIter1 , typename _FIter2 >
  void **std::iter_swap** (_FIter1, _FIter2)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _Compare >
  bool **std::lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _FIter , typename _Tp >
  _FIter **std::lower_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  _FIter **std::lower_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _RAIter >
  void **std::make_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::make_heap** (_RAIter, _RAIter, _Compare)

- template<typename _Tp >
  const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  _Tp **std::max** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  _Tp **std::max** (initializer_list< _Tp >, _Compare)
- template<typename _FIter >
  _FIter **std::max_element** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  _FIter **std::max_element** (_FIter, _FIter, _Compare)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _Tp >
  const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  _Tp **std::min** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  _Tp **std::min** (initializer_list< _Tp >, _Compare)
- template<typename _FIter >
  _FIter **std::min_element** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  _FIter **std::min_element** (_FIter, _FIter, _Compare)
- template<typename _Tp >
  pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >, _Compare)
- template<typename _FIter >
  pair< _FIter, _FIter > **std::minmax_element** (_FIter, _FIter)
- template<typename _FIter , typename _Compare >
  pair< _FIter, _FIter > **std::minmax_element** (_FIter, _FIter, _Compare)
- template<typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **std::mismatch** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
  pair< _IIter1, _IIter2 > **std::mismatch** (_IIter1, _IIter1, _IIter2, _BinaryPredicate)
- template<typename _BIter >
  bool **std::next_permutation** (_BIter, _BIter)
- template<typename _BIter , typename _Compare >
  bool **std::next_permutation** (_BIter, _BIter, _Compare)
- template<typename _IIter , typename _Predicate >
  bool **std::none_of** (_IIter, _IIter, _Predicate)
- template<typename _RAIter >
  void **std::nth_element** (_RAIter, _RAIter, _RAIter)

- template<typename _RAIter , typename _Compare >
  void **std::nth_element** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RAIter >
  void **std::partial_sort** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::partial_sort** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _IIter , typename _RAIter >
  _RAIter **std::partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter)
- template<typename _IIter , typename _RAIter , typename _Compare >
  _RAIter **std::partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter, _Compare)
- template<typename _BIter , typename _Predicate >
  _BIter **std::partition** (_BIter, _BIter, _Predicate)
- template<typename _IIter , typename _OIter1 , typename _OIter2 , typename _Predicate >
  pair< _OIter1, _OIter2 > **std::partition_copy** (_IIter, _IIter, _OIter1, _OIter2, _Predicate)
- template<typename _FIter , typename _Predicate >
  _FIter **std::partition_point** (_FIter, _FIter, _Predicate)
- template<typename _RAIter >
  void **std::pop_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::pop_heap** (_RAIter, _RAIter, _Compare)
- template<typename _BIter >
  bool **std::prev_permutation** (_BIter, _BIter)
- template<typename _BIter , typename _Compare >
  bool **std::prev_permutation** (_BIter, _BIter, _Compare)
- template<typename _RAIter >
  void **std::push_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::push_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter >
  void **std::random_shuffle** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Generator >
  void **std::random_shuffle** (_RAIter, _RAIter, _Generator &&)
- template<typename _FIter , typename _Tp >
  _FIter **std::remove** (_FIter, _FIter, const _Tp &)
- template<typename _IIter , typename _OIter , typename _Tp >
  _OIter **std::remove_copy** (_IIter, _IIter, _OIter, const _Tp &)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **std::remove_copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _FIter , typename _Predicate >
  _FIter **std::remove_if** (_FIter, _FIter, _Predicate)
- template<typename _FIter , typename _Tp >
  void **std::replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _IIter , typename _OIter , typename _Tp >
  _OIter **std::replace_copy** (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)
- template<typename _Iter , typename _OIter , typename _Predicate , typename _Tp >
  _OIter **std::replace_copy_if** (_Iter, _Iter, _OIter, _Predicate, const _Tp &)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **std::replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _BIter >
  void **std::reverse** (_BIter, _BIter)
- template<typename _BIter , typename _OIter >
  _OIter **std::reverse_copy** (_BIter, _BIter, _OIter)

- template<typename _FIter >
  void **std::rotate** (_FIter, _FIter, _FIter)
- template<typename _FIter , typename _OIter >
  _OIter **std::rotate_copy** (_FIter, _FIter, _FIter, _OIter)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **std::search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BinaryPredicate >
  _FIter1 **std::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _FIter , typename _Size , typename _Tp >
  _FIter **std::search_n** (_FIter, _FIter, _Size, const _Tp &)
- template<typename _FIter , typename _Size , typename _Tp , typename _BinaryPredicate >
  _FIter **std::search_n** (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _RAIter , typename _UGenerator >
  void **std::shuffle** (_RAIter, _RAIter, _UGenerator &&)
- template<typename _RAIter >
  void **std::sort** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::sort** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter >
  void **std::sort_heap** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::sort_heap** (_RAIter, _RAIter, _Compare)
- template<typename _BIter , typename _Predicate >
  _BIter **std::stable_partition** (_BIter, _BIter, _Predicate)
- template<typename _RAIter >
  void **std::stable_sort** (_RAIter, _RAIter)
- template<typename _RAIter , typename _Compare >
  void **std::stable_sort** (_RAIter, _RAIter, _Compare)
- template<typename _Tp >
  void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >
- template<typename _Tp , size_t _Nm>
  void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(∗__a
- template<typename _FIter1 , typename _FIter2 >
  _FIter2 **std::swap_ranges** (_FIter1, _FIter1, _FIter2)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **std::transform** (_IIter, _IIter, _OIter, _UnaryOperation)

- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BinaryOperation >
  _OIter **std::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)
- template<typename _FIter >
  _FIter **std::unique** (_FIter, _FIter)
- template<typename _FIter , typename _BinaryPredicate >
  _FIter **std::unique** (_FIter, _FIter, _BinaryPredicate)
- template<typename _IIter , typename _OIter >
  _OIter **std::unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter , typename _OIter , typename _BinaryPredicate >
  _OIter **std::unique_copy** (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _FIter , typename _Tp >
  _FIter **std::upper_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter , typename _Tp , typename _Compare >
  _FIter **std::upper_bound** (_FIter, _FIter, const _Tp &, _Compare)

Variables

- void $*$ **std::__b**

5.6.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

Definition in file bits/algorithmfwd.h.

5.7 algorithmfwd.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _FIter , typename _IterTag >
  _FIter **std::__parallel::__adjacent_find_switch** (_FIter, _FIter, _IterTag)
- template<typename _FIter , typename _BiPredicate , typename _IterTag >
  _FIter **std::__parallel::__adjacent_find_switch** (_FIter, _FIter, _BiPredicate, _IterTag)
- template<typename _RAIter , typename _BiPredicate >
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)
- template<typename _RAIter >
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _IIter , typename _Predicate , typename _IterTag >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::__count_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter , typename _Predicate >
  iterator_traits< _RAIter >
  ::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)

- template<typename _IIter , typename _Tp , typename _IterTag >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::__count_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter , typename _Tp >
  iterator_traits< _RAIter >
  ::difference_type **std::__parallel::__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _FIter , typename _IterTag1 , typename _IterTag2 >
  _IIter **std::__parallel::__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)
- template<typename _RAIter , typename _FIter , typename _BiPredicate , typename _IterTag >
  _RAIter **std::__parallel::__find_first_of_switch** (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)
- template<typename _IIter , typename _FIter , typename _BiPredicate , typename _IterTag1 , typename _IterTag2 >
  _IIter **std::__parallel::__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _IIter , typename _Predicate , typename _IterTag >
  _IIter **std::__parallel::__find_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter , typename _Predicate >
  _RAIter **std::__parallel::__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _RAIter , typename _Tp >
  _RAIter **std::__parallel::__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter , typename _Tp , typename _IterTag >
  _IIter **std::__parallel::__find_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter , typename _Function >
  _Function **std::__parallel::__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _IIter , typename _Function , typename _IterTag >
  _Function **std::__parallel::__for_each_switch** (_IIter, _IIter, _Function, _IterTag)
- template<typename _OIter , typename _Size , typename _Generator , typename _IterTag >
  _OIter **std::__parallel::__generate_n_switch** (_OIter, _Size, _Generator, _IterTag)
- template<typename _RAIter , typename _Size , typename _Generator >
  _RAIter **std::__parallel::__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIter , typename _Generator , typename _IterTag >
  void **std::__parallel::__generate_switch** (_FIter, _FIter, _Generator, _IterTag)
- template<typename _RAIter , typename _Generator >
  void **std::__parallel::__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _Predicate >
  bool **std::__parallel::__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IterTag1 , typename _IterTag2 >
  bool **std::__parallel::__lexicographical_compare_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _FIter , typename _Compare , typename _IterTag >
  _FIter **std::__parallel::__max_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _RAIter , typename _Compare >
  _RAIter **std::__parallel::__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)

- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **std::__parallel::__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::__parallel::__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter , typename _Compare , typename _IterTag >
  _FIter **std::__parallel::__min_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _RAIter , typename _Compare >
  _RAIter **std::__parallel::__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _Predicate >
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _IterTag1 , typename _IterTag2 >
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _FIter , typename _Predicate , typename _IterTag >
  _FIter **std::__parallel::__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _RAIter , typename _Predicate >
  _RAIter **std::__parallel::__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIter , typename _Predicate , typename _Tp , typename _IterTag >
  void **std::__parallel::__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)
- template<typename _RAIter , typename _Predicate , typename _Tp >
  void **std::__parallel::__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _FIter , typename _Tp , typename _IterTag >
  void **std::__parallel::__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _RAIter , typename _Tp >
  void **std::__parallel::__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter , typename _Integer , typename _Tp , typename _BiPredicate >
  _RAIter **std::__parallel::__search_n_switch** (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)
- template<typename _FIter , typename _Integer , typename _Tp , typename _BiPredicate , typename _IterTag >
  _FIter **std::__parallel::__search_n_switch** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)
- template<typename _FIter1 , typename _FIter2 , typename _IterTag1 , typename _IterTag2 >
  _FIter1 **std::__parallel::__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)
- template<typename _RAIter1 , typename _RAIter2 , typename _BiPredicate >
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter1 , typename _FIter2 , typename _BiPredicate , typename _IterTag1 , typename _IterTag2 >
  _FIter1 **std::__parallel::__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _RAIter1 , typename _RAIter2 >
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **std::__parallel::__set_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **std::__parallel::__set_intersection_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **std::__parallel::__set_symmetric_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1 , typename _RAIter2 , typename _Output_RAIter , typename _Predicate >
  _Output_RAIter **std::__parallel::__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate , typename _OIter , typename _IterTag1 , typename _IterTag2 , typename _IterTag3 >
  _OIter **std::__parallel::__set_union_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter , typename _OIter , typename _UnaryOperation , typename _IterTag1 , typename _IterTag2 >
  _OIter **std::__parallel::__transform1_switch** (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
- template<typename _RAIIter , typename _RAOIter , typename _UnaryOperation >
  _RAOIter **std::__parallel::__transform1_switch** (_RAIIter, _RAIIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _BiOperation >
  _RAIter3 **std::__parallel::__transform2_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation , typename _Tag1 , typename _Tag2 , typename _Tag3 >
  _OIter **std::__parallel::__transform2_switch** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)
- template<typename _IIter , typename _OIter , typename _Predicate , typename _IterTag1 , typename _IterTag2 >
  _OIter **std::__parallel::__unique_copy_switch** (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter , typename _RandomAccess_OIter , typename _Predicate >
  _RandomAccess_OIter **std::__parallel::__unique_copy_switch** (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter >
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter)
- template<typename _FIter >
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _BiPredicate >
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, _BiPredicate)

- template<typename _FIter , typename _BiPredicate >
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Tp >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Predicate >
  iterator_traits< _IIter >
  ::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter , typename _Tp >
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter , typename _FIter >
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIter , typename _BiPredicate >
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _FIter , typename _BiPredicate >
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate)
- template<typename _IIter , typename _FIter >
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter)
- template<typename _IIter , typename _Predicate >
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Predicate >
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter , typename _Function >
  _Function **std::__parallel::for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)

- template<typename _Iterator , typename _Function >
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Function >
  _Function **std::__parallel::for_each** (_IIter, _IIter, _Function)
- template<typename _FIter , typename _Generator >
  void **std::__parallel::generate** (_FIter, _FIter, _Generator)
- template<typename _FIter , typename _Generator >
  void **std::__parallel::generate** (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Generator >
  void **std::__parallel::generate** (_FIter, _FIter, _Generator, __gnu_parallel::_Parallelism)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)
- template<typename _OIter , typename _Size , typename _Generator >
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _FIter >
  _FIter **std::__parallel::max_element** (_FIter, _FIter)
- template<typename _FIter >
  _FIter **std::__parallel::max_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter >
  _FIter **std::__parallel::max_element** (_FIter, _FIter, __gnu_parallel::_Parallelism)
- template<typename _FIter , typename _Compare >
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter , typename _Compare >
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Compare >
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Compare >
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _FIter >
  _FIter **std::__parallel::min_element** (_FIter, _FIter)

- template<typename _FIter >
  _FIter **std::__parallel::min_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter >
  _FIter **std::__parallel::min_element** (_FIter, _FIter, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter , typename _Compare >
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare)
- template<typename _FIter , typename _Compare >
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Compare >
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1 , typename _IIter2 , typename _Predicate >
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _RAIter >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _FIter , typename _Predicate >
  _FIter **std::__parallel::partition** (_FIter, _FIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Predicate >
  _FIter **std::__parallel::partition** (_FIter, _FIter, _Predicate)
- template<typename _RAIter >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end)

- template<typename _RAIter , typename _RandomNumberGenerator >
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _FIter , typename _Tp >
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _FIter , typename _Tp >
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Tp >
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Predicate , typename _Tp >
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, __gnu_parallel::sequential_tag)
- template<typename _FIter1 , typename _FIter2 >
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1 , typename _FIter2 , typename _BiPredicate >
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIter1 , typename _FIter2 , typename _BiPredicate >
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)
- template<typename _FIter , typename _Integer , typename _Tp >
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Integer , typename _Tp , typename _BiPredicate >
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIter , typename _Integer , typename _Tp >
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &)
- template<typename _FIter , typename _Integer , typename _Tp , typename _BiPredicate >
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)

- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter >
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter , typename _Compare >
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter , typename _UnaryOperation >
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation >
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation >
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _OIter , typename _BiOperation >
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter , typename _OIter , typename _Predicate >
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, _Predicate)

5.7.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file parallel/algorithmfwd.h.

5.8 alloc_traits.h File Reference

Classes

- struct std::allocator_traits< _Alloc >

  Uniform interface to all allocator types.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ALLOC_TR_NESTED_TYPE**(_NTYPE, _ALT)

Typedefs

- template<typename _Alloc >
  using **std::__check_copy_constructible** = __allow_copy_cons< __is_copy_insertable< _Alloc >::value >

Functions

- template<typename _Alloc >
  void **std::__alloc_on_copy** (_Alloc &__one, const _Alloc &__two)
- template<typename _Alloc >
  _Alloc **std::__alloc_on_copy** (const _Alloc &__a)
- template<typename _Alloc >
  void **std::__alloc_on_move** (_Alloc &__one, _Alloc &__two)
- template<typename _Alloc >
  void **std::__alloc_on_swap** (_Alloc &__one, _Alloc &__two)
- template<typename _Alloc >
  void **std::__do_alloc_on_copy** (_Alloc &__one, const _Alloc &__two, true_type)
- template<typename _Alloc >
  void **std::__do_alloc_on_move** (_Alloc &__one, _Alloc &__two, true_type)
- template<typename _Alloc >
  void **std::__do_alloc_on_swap** (_Alloc &__one, _Alloc &__two, true_type)

5.8.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file bits/alloc_traits.h.

5.9 alloc_traits.h File Reference

Classes

- struct __gnu_cxx::__alloc_traits< _Alloc >

  Uniform interface to C++98 and C++0x allocators.
- struct std::allocator< _Tp >

  The standard allocator, as per [20.4].

Namespaces

- namespace __gnu_cxx
- namespace std

5.9.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file ext/alloc_traits.h.

5.10 allocator.h File Reference

Classes

- struct std::allocator< _Tp >

  The standard allocator, as per [20.4].
- class std::allocator< void >

  allocator<void> specialization.
- struct uses_allocator< typename, typename >

  Declare uses_allocator so it can be specialized in <queue> etc.

Namespaces

- namespace std

Functions

- template<typename _T1 , typename _T2 >
  bool **std::operator!=** (const allocator< _T1 > &, const allocator< _T2 > &)
- template<typename _Tp >
  bool **std::operator!=** (const allocator< _Tp > &, const allocator< _Tp > &)
- template<typename _T1 , typename _T2 >
  bool **std::operator==** (const allocator< _T1 > &, const allocator< _T2 > &)
- template<typename _Tp >
  bool **std::operator==** (const allocator< _Tp > &, const allocator< _Tp > &)

5.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file allocator.h.

5.11 array File Reference

Classes

- struct std::array< _Tp, _Nm >

  A standard container for storing a fixed size sequence of elements.
- class tuple_element< _Int, _Tp >

  tuple_element
- class tuple_size< _Tp >

  tuple_size

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ARRAY**

Functions

- template<std::size_t _Int, typename _Tp , std::size_t _Nm>
  constexpr _Tp & **std::get** (array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t _Int, typename _Tp , std::size_t _Nm>
  constexpr _Tp && **std::get** (array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp , std::size_t _Nm>
  constexpr const _Tp & **std::get** (const array< _Tp, _Nm > &__arr) noexcept
- template<typename _Tp , std::size_t _Nm>
  bool **std::operator!=** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , std::size_t _Nm>
  bool **std::operator<** (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)
- template<typename _Tp , std::size_t _Nm>
  bool **std::operator<=** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , std::size_t _Nm>
  bool **std::operator==** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , std::size_t _Nm>
  bool **std::operator>** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , std::size_t _Nm>
  bool **std::operator>=** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp , std::size_t _Nm>
  void **std::swap** (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))

5.11.1 Detailed Description

This is a Standard C++ Library header.

Definition in file array.

5.12 array_allocator.h File Reference

Classes

- class __gnu_cxx::array_allocator< _Tp, _Array >

  An allocator that uses previously allocated memory. This memory can be externally, globally, or otherwise allocated.
- class __gnu_cxx::array_allocator_base< _Tp >

  Base class.

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp , typename _Array >
  bool **__gnu_cxx::operator!=** (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)
- template<typename _Tp , typename _Array >
  bool **__gnu_cxx::operator==** (const array_allocator< _Tp, _Array > &, const array_allocator< _Tp, _Array > &)

5.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file array_allocator.h.

5.13 assoc_container.hpp File Reference

Classes

- class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
- class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
- class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
- class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_CC_HASH_BASE**
- #define **PB_DS_GP_HASH_BASE**
- #define **PB_DS_HASH_BASE**
- #define **PB_DS_LU_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

5.13.1 Detailed Description

Contains associative containers.

Definition in file assoc_container.hpp.

5.14 atomic File Reference

Classes

- struct std::atomic< _Tp >

  Generic atomic type, primary class template.
- struct std::atomic< _Tp ∗ >

  Partial specialization for pointer types.
- struct std::atomic< bool >

  Explicit specialization for bool.
- struct std::atomic< char >

  Explicit specialization for char.
- struct std::atomic< char16_t >

  Explicit specialization for char16_t.
- struct std::atomic< char32_t >

  Explicit specialization for char32_t.
- struct std::atomic< int >

  Explicit specialization for int.
- struct std::atomic< long >

  Explicit specialization for long.
- struct std::atomic< long long >

  Explicit specialization for long long.
- struct std::atomic< short >

  Explicit specialization for short.
- struct std::atomic< signed char >

  Explicit specialization for signed char.
- struct std::atomic< unsigned char >

  Explicit specialization for unsigned char.
- struct std::atomic< unsigned int >

  Explicit specialization for unsigned int.

- struct std::atomic< unsigned long >

  Explicit specialization for unsigned long.
- struct std::atomic< unsigned long long >

  Explicit specialization for unsigned long long.
- struct std::atomic< unsigned short >

  Explicit specialization for unsigned short.
- struct std::atomic< wchar_t >

  Explicit specialization for wchar_t.
- struct std::atomic_bool

  atomic_bool

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ATOMIC**

Functions

- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak** (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak_explicit** (atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  bool **std::atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > *__a, _ITp *__i1, _ITp __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange** (atomic< _ITp > *__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange** (volatile atomic< _ITp > *__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange_explicit** (atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_exchange_explicit** (volatile atomic< _ITp > *__a, _ITp __i, memory_order __m) noexcept

- template<typename _ITp >
  _ITp **std::atomic_fetch_add** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_add_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add_explicit** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_add_explicit** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_sub_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub_explicit** (volatile atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept

- template<typename _ITp >
  _ITp ∗ **std::atomic_fetch_sub_explicit** (atomic< _ITp ∗ > ∗__a, ptrdiff_t __d, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor** (__atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor** (volatile __atomic_base< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor_explicit** (__atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- void **std::atomic_flag_clear** (atomic_flag ∗__a) noexcept
- void **std::atomic_flag_clear** (volatile atomic_flag ∗__a) noexcept
- void **std::atomic_flag_clear_explicit** (atomic_flag ∗__a, memory_order __m) noexcept
- void **std::atomic_flag_clear_explicit** (volatile atomic_flag ∗__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set** (atomic_flag ∗__a) noexcept
- bool **std::atomic_flag_test_and_set** (volatile atomic_flag ∗__a) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (atomic_flag ∗__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (volatile atomic_flag ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  void **std::atomic_init** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **std::atomic_init** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  bool **std::atomic_is_lock_free** (const atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  bool **std::atomic_is_lock_free** (const volatile atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load** (const atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load** (const volatile atomic< _ITp > ∗__a) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load_explicit** (const atomic< _ITp > ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  _ITp **std::atomic_load_explicit** (const volatile atomic< _ITp > ∗__a, memory_order __m) noexcept
- template<typename _ITp >
  void **std::atomic_store** (atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **std::atomic_store** (volatile atomic< _ITp > ∗__a, _ITp __i) noexcept
- template<typename _ITp >
  void **std::atomic_store_explicit** (atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept
- template<typename _ITp >
  void **std::atomic_store_explicit** (volatile atomic< _ITp > ∗__a, _ITp __i, memory_order __m) noexcept

5.14.1 Detailed Description

This is a Standard C++ Library header.

Definition in file atomic.

5.15 atomic_base.h File Reference

Classes

- struct std::__atomic_base< _ITp >

  Base class for atomic integrals.
- struct std::__atomic_base< _PTp ∗ >

  Partial specialization for pointer types.
- struct std::__atomic_flag_base

  Base type for atomic_flag.
- struct std::atomic< _Tp >

  Generic atomic type, primary class template.
- struct std::atomic_flag

  atomic_flag

Namespaces

- namespace std

Macros

- #define **ATOMIC_FLAG_INIT**
- #define **ATOMIC_VAR_INIT**(_VI)

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**
- typedef __atomic_base< char > std::atomic_char
- typedef __atomic_base< char16_t > std::atomic_char16_t
- typedef __atomic_base< char32_t > std::atomic_char32_t
- typedef __atomic_base< int > std::atomic_int
- typedef __atomic_base < int_fast16_t > std::atomic_int_fast16_t
- typedef __atomic_base < int_fast32_t > std::atomic_int_fast32_t
- typedef __atomic_base < int_fast64_t > std::atomic_int_fast64_t
- typedef __atomic_base < int_fast8_t > std::atomic_int_fast8_t
- typedef __atomic_base < int_least16_t > std::atomic_int_least16_t
- typedef __atomic_base < int_least32_t > std::atomic_int_least32_t
- typedef __atomic_base < int_least64_t > std::atomic_int_least64_t
- typedef __atomic_base < int_least8_t > std::atomic_int_least8_t
- typedef __atomic_base< intmax_t > std::atomic_intmax_t
- typedef __atomic_base< intptr_t > std::atomic_intptr_t
- typedef __atomic_base< long long > std::atomic_llong

- typedef __atomic_base< long > std::atomic_long
- typedef __atomic_base< ptrdiff_t > std::atomic_ptrdiff_t
- typedef __atomic_base< signed char > std::atomic_schar
- typedef __atomic_base< short > std::atomic_short
- typedef __atomic_base< size_t > std::atomic_size_t
- typedef __atomic_base < unsigned char > std::atomic_uchar
- typedef __atomic_base < unsigned int > std::atomic_uint
- typedef __atomic_base < uint_fast16_t > std::atomic_uint_fast16_t
- typedef __atomic_base < uint_fast32_t > std::atomic_uint_fast32_t
- typedef __atomic_base < uint_fast64_t > std::atomic_uint_fast64_t
- typedef __atomic_base < uint_fast8_t > std::atomic_uint_fast8_t
- typedef __atomic_base < uint_least16_t > std::atomic_uint_least16_t
- typedef __atomic_base < uint_least32_t > std::atomic_uint_least32_t
- typedef __atomic_base < uint_least64_t > std::atomic_uint_least64_t
- typedef __atomic_base < uint_least8_t > std::atomic_uint_least8_t
- typedef __atomic_base< uintmax_t > std::atomic_uintmax_t
- typedef __atomic_base< uintptr_t > std::atomic_uintptr_t
- typedef __atomic_base < unsigned long long > std::atomic_ullong
- typedef __atomic_base < unsigned long > std::atomic_ulong
- typedef __atomic_base < unsigned short > std::atomic_ushort
- typedef __atomic_base< wchar_t > std::atomic_wchar_t
- typedef enum std::memory_order std::memory_order

Enumerations

- enum **__memory_order_modifier** { **__memory_order_mask**, **__memory_order_modifier_mask**, **__memory_order_hle_acquire**, **__memory_order_hle_release** }
- enum std::memory_order { **memory_order_relaxed**, **memory_order_consume**, **memory_order_acquire**, **memory_order_release**, **memory_order_acq_rel**, **memory_order_seq_cst** }

Functions

- constexpr memory_order **std::__cmpexch_failure_order** (memory_order __m) noexcept
- constexpr memory_order **std::__cmpexch_failure_order2** (memory_order __m) noexcept
- void **std::atomic_signal_fence** (memory_order __m) noexcept
- void **std::atomic_thread_fence** (memory_order __m) noexcept

- template<typename _Tp >
  _Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order **std::operator&** (memory_order __m, __memory_order_modifier __mod)
- constexpr memory_order **std::operator|** (memory_order __m, __memory_order_modifier __mod)

5.15.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

Definition in file atomic_base.h.

5.16 atomic_lockfree_defines.h File Reference

Macros

- #define ATOMIC_BOOL_LOCK_FREE
- #define **ATOMIC_CHAR16_T_LOCK_FREE**
- #define **ATOMIC_CHAR32_T_LOCK_FREE**
- #define **ATOMIC_CHAR_LOCK_FREE**
- #define **ATOMIC_INT_LOCK_FREE**
- #define **ATOMIC_LLONG_LOCK_FREE**
- #define **ATOMIC_LONG_LOCK_FREE**
- #define **ATOMIC_POINTER_LOCK_FREE**
- #define **ATOMIC_SHORT_LOCK_FREE**
- #define **ATOMIC_WCHAR_T_LOCK_FREE**

5.16.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

Definition in file atomic_lockfree_defines.h.

5.17 atomic_word.h File Reference

Typedefs

- typedef int **_Atomic_word**

5.17.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file atomic_word.h.

5.18 atomicity.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_GLIBCXX_READ_MEM_BARRIER**
- #define **_GLIBCXX_WRITE_MEM_BARRIER**

Functions

- static void **__gnu_cxx::__atomic_add_single** (_Atomic_word ∗__mem, int __val)
- else **__gnu_cxx::__atomic_add_single** (__mem, __val)
- _Atomic_word **__gnu_cxx::__attribute__** ((__unused__)) __exchange_and_add(volatile _Atomic_word ∗
- static _Atomic_word **__gnu_cxx::__exchange_and_add_single** (_Atomic_word ∗__mem, int __val)
- else return **__gnu_cxx::__exchange_and_add_single** (__mem, __val)
- static _Atomic_word int __val **__gnu_cxx::if** (__gthread_active_p()) return __exchange_and_add(__mem
- _Atomic_word int **__gnu_cxx::throw** ()

Variables

- static _Atomic_word int __val **__gnu_cxx::__val**

5.18.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file atomicity.h.

5.19 auto_ptr.h File Reference

Classes

- class std::auto_ptr< _Tp >

  A simple smart pointer providing strict ownership semantics.
- struct std::auto_ptr_ref< _Tp1 >

Namespaces

- namespace std

5.19.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file auto_ptr.h.

5.20 backward_warning.h File Reference

5.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

Definition in file backward_warning.h.

5.21 balanced_quicksort.h File Reference

Classes

- struct __gnu_parallel::_QSBThreadLocal< _RAIter >

  Information local to one thread in the parallel quicksort run.

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__qsb_conquer (_QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter , typename _Compare >
  std::iterator_traits< _RAIter >
  ::difference_type __gnu_parallel::__qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)

5.21.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort. It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file balanced_quicksort.h.

5.22 base.h File Reference

Namespaces

- namespace __gnu_profile
- namespace std
- namespace std::__profile

5.22.1 Detailed Description

Sequential helper functions. This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/base.h.

5.23 base.h File Reference

Classes

- class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

  Similar to std::binder1st, but giving the argument types explicitly.
- class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

  Similar to std::binder2nd, but giving the argument types explicitly.
- class __gnu_parallel::__unary_negate< _Predicate, argument_type >

  Similar to std::unary_negate, but giving the argument types explicitly.
- class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >

  Constructs predicate for equality from strict weak ordering predicate.
- struct __gnu_parallel::_EqualTo< _T1, _T2 >

  Similar to std::equal_to, but allows two different types.
- struct __gnu_parallel::_Less< _T1, _T2 >

  Similar to std::less, but allows two different types.
- struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >

  Similar to std::multiplies, but allows two different types.
- struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >

  Similar to std::plus, but allows two different types.
- class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >

  Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.
- class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >

  _Iterator associated with __gnu_parallel::_PseudoSequence. If features the usual random-access iterator functionality.

Namespaces

- namespace __gnu_parallel
- namespace __gnu_sequential
- namespace std
- namespace std::__parallel

Macros

- #define **_GLIBCXX_PARALLEL_ASSERT**(_Condition)

Functions

- void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b)
- _CASable __gnu_parallel::__encode2 (int __a, int __b)
- _ThreadIndex **__gnu_parallel::__get_max_threads** ()
- bool **__gnu_parallel::__is_parallel** (const _Parallelism __p)
- template<typename _RAIter , typename _Compare >
  _RAIter __gnu_parallel::__median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)
- template<typename _Size >
  _Size __gnu_parallel::__rd_log2 (_Size __n)
- template<typename _Tp >
  const _Tp & __gnu_parallel::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp >
  const _Tp & __gnu_parallel::min (const _Tp &__a, const _Tp &__b)

5.23.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file parallel/base.h.

5.24 basic_file.h File Reference

Namespaces

- namespace std

5.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

Definition in file basic_file.h.

5.25 basic_ios.h File Reference

Classes

- class std::basic_ios< _CharT, _Traits >

  Template class basic_ios, virtual base class for all stream classes.

Namespaces

- namespace std

Functions

- template<typename _Facet >
  const _Facet & **std::__check_facet** (const _Facet *__f)

5.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

Definition in file basic_ios.h.

5.26 basic_ios.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_BASIC_IOS_TCC**

5.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

Definition in file basic_ios.tcc.

5.27 basic_iterator.h File Reference

5.27.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file basic_iterator.h.

5.28 basic_string.h File Reference

Classes

- class std::basic_string< _CharT, _Traits, _Alloc >

  Managing sequences of characters and character-like objects.
- struct std::hash< string >

  std::hash specialization for string.
- struct std::hash< u16string >

  std::hash specialization for u16string.
- struct std::hash< u32string >

  std::hash specialization for u32string.
- struct std::hash< wstring >

  std::hash specialization for wstring.

Namespaces

- namespace std

Functions

- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<>
  basic_istream< char > & **std::getline** (basic_istream< char > &__in, basic_string< char > &__str, char __delim)
- template<>
  basic_istream< wchar_t > & **std::getline** (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator!= (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > **std::operator+** (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > **std::operator+** (const _CharT ∗__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > **std::operator+** (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits,
  _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator< (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator<= (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  bool >::__type **std::operator==** (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator== (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator> (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT ∗__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc >
  bool std::operator>= (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<>
  basic_istream< char > & **std::operator>>** (basic_istream< char > &__is, basic_string< char > &__str)
- template<typename _CharT , typename _Traits , typename _Alloc >
  void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)

5.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

Definition in file basic_string.h.

5.29 basic_string.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_BASIC_STRING_TCC**

Functions

- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)

5.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

Definition in file basic_string.tcc.

5.30 bin_search_tree_.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node)
- #define **PB_DS_BIN_TREE_NAME**
- #define **PB_DS_BIN_TREE_TRAITS_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_STRUCT_ONLY_ASSERT_VALID**(X)

5.30.1 Detailed Description

Contains an implementation class for binary search tree.

Definition in file bin_search_tree_.hpp.

5.31 binary_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_DEBUG_VERIFY**(_Cond)
- #define **PB_DS_ENTRY_CMP_DEC**
- #define **PB_DS_RESIZE_POLICY_DEC**

5.31.1 Detailed Description

Contains an implementation class for a binary heap.

Definition in file binary_heap_.hpp.

5.32 binders.h File Reference

Classes

- class std::binder1st< _Operation >

  One of the binder functors.
- class std::binder2nd< _Operation >

  One of the binder functors.

Namespaces

- namespace std

Functions

- template<typename _Operation , typename _Tp >
  binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation , typename _Tp >
  binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)

5.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

Definition in file binders.h.

5.33 binomial_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.33.1 Detailed Description

Contains an implementation class for a binomial heap.

Definition in file binomial_heap_.hpp.

5.34 binomial_heap_base_.hpp File Reference

Classes

- class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >

  Base class for binomial heap.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_ASSERT_VALID_COND**(X, _StrictlyBinomial)
- #define **PB_DS_B_HEAP_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.34.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_.hpp.

5.35 bitmap_allocator.h File Reference

Classes

- class __gnu_cxx::__detail::__mini_vector< _Tp >

  __mini_vector<> is a stripped down version of the full-fledged std::vector<>.
- class __gnu_cxx::__detail::_Bitmap_counter< _Tp >

  The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.
- class __gnu_cxx::__detail::_Ffit_finder< _Tp >

  The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.
- class __gnu_cxx::bitmap_allocator< _Tp >

  Bitmap Allocator, primary template.
- class __gnu_cxx::free_list

  The free list class for managing chunks of memory to be given to and returned by the bitmap_allocator.

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::__detail

Macros

- #define _BALLOC_ALIGN_BYTES

Enumerations

- enum { **bits_per_byte**, **bits_per_block** }

Functions

- void __gnu_cxx::__detail::__bit_allocate (size_t ∗__pbmap, size_t __pos) throw ()
- void __gnu_cxx::__detail::__bit_free (size_t ∗__pbmap, size_t __pos) throw ()
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator **__gnu_cxx::__detail::__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair >
  size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair __ap)
- template<typename _AddrPair >
  size_t __gnu_cxx::__detail::__num_blocks (_AddrPair __ap)
- size_t __gnu_cxx::_Bit_scan_forward (size_t __num)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator!=** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator==** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()

5.35.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file bitmap_allocator.h.

5.35.2 Macro Definition Documentation

5.35.2.1 #define _BALLOC_ALIGN_BYTES

The constant in the expression below is the alignment required in bytes.

Definition at line 43 of file bitmap_allocator.h.

5.36 bitset File Reference

Classes

- struct std::_Base_bitset< _Nw >
- struct std::_Base_bitset< 0 >
- struct std::_Base_bitset< 1 >
- struct std::hash<::bitset< _Nb > >

  std::hash specialization for bitset.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_BITSET**
- #define **_GLIBCXX_BITSET_BITS_PER_ULL**
- #define **_GLIBCXX_BITSET_BITS_PER_WORD**
- #define **_GLIBCXX_BITSET_WORDS**(__n)

Functions

- size_t std::_Find_first () const noexcept
- size_t std::_Find_next (size_t __prev) const noexcept
- template<class _CharT , class _Traits >
  void **std::_M_copy_from_ptr** (const _CharT ∗, size_t, size_t, size_t, _CharT, _CharT)
- template<class _CharT , class _Traits , class _Alloc >
  void **std::_M_copy_from_string** (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n, _CharT __zero, _CharT __one)
- template<class _CharT , class _Traits , class _Alloc >
  void **std::_M_copy_from_string** (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __pos, size_t __n)
- template<class _CharT , class _Traits , class _Alloc >
  void **std::_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc > &, _CharT, _CharT) const
- template<class _CharT , class _Traits , class _Alloc >
  void **std::_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc > &__s) const
- template<size_t _Nb>
  std::_M_do_and (__rhs)
- bool std::all () const noexcept
- bool std::any () const noexcept
- size_t std::count () const noexcept
- bitset< _Nb > & std::flip () noexcept
- bitset< _Nb > & std::flip (size_t __position)
- bool std::none () const noexcept
- bitset< _Nb > & **std::operator^=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > & **std::operator|=** (const bitset< _Nb > &__rhs) noexcept
- bitset< _Nb > std::operator~ () const noexcept
- bitset< _Nb > & std::reset () noexcept
- bitset< _Nb > & std::reset (size_t __position)
- bitset< _Nb > & std::set () noexcept
- bitset< _Nb > & std::set (size_t __position, bool __val=true)
- constexpr size_t std::size () const noexcept
- bool std::test (size_t __position) const
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_string< _CharT, _Traits, _Alloc > std::to_string () const
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_string< _CharT, _Traits, _Alloc > **std::to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT , class _Traits >
  std::basic_string< _CharT, _Traits, std::allocator < _CharT > > **std::to_string** () const

- template<class _CharT , class _Traits >
  std::basic_string< _CharT,
  _Traits, std::allocator
  < _CharT > > **std::to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **std::to_string** () const
- template<class _CharT >
  std::basic_string< _CharT,
  std::char_traits< _CharT >
  , std::allocator< _CharT > > **std::to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- std::basic_string< char,
  std::char_traits< char >
  , std::allocator< char > > **std::to_string** (char __zero, char __one= '1') const
- unsigned long long **std::to_ullong** () const
- unsigned long std::to_ulong () const

- bitset< _Nb > & std::operator<<= (size_t __position) noexcept
- bitset< _Nb > & std::operator>>= (size_t __position) noexcept

- bitset< _Nb > & std::_Unchecked_set (size_t __pos) noexcept
- bitset< _Nb > & std::_Unchecked_set (size_t __pos, int __val) noexcept
- bitset< _Nb > & std::_Unchecked_reset (size_t __pos) noexcept
- bitset< _Nb > & std::_Unchecked_flip (size_t __pos) noexcept
- constexpr bool std::_Unchecked_test (size_t __pos) const noexcept

- reference std::operator[] (size_t __position)

- bool std::operator== (const bitset< _Nb > &__rhs) const noexcept
- bool std::operator!= (const bitset< _Nb > &__rhs) const noexcept

- bitset< _Nb > std::operator<< (size_t __position) const noexcept
- bitset< _Nb > std::operator>> (size_t __position) const noexcept

- template<size_t _Nb>
  bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

- template<class _CharT , class _Traits , size_t _Nb>
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<class _CharT , class _Traits , size_t _Nb>
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)

Variables

- return ∗ **std::this**

5.36.1 Detailed Description

This is a Standard C++ Library header.

Definition in file bitset.

5.37 bitset File Reference

Classes

- class std::__debug::bitset< _Nb >

  Class std::bitset with additional safety/checking/debug instrumentation.
- struct std::hash< __debug::bitset< _Nb > >

  std::hash specialization for bitset.

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<size_t _Nb>
  bitset< _Nb > **std::__debug::operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_ostream< _CharT,
  _Traits > & **std::__debug::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_istream< _CharT,
  _Traits > & **std::__debug::operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  bitset< _Nb > **std::__debug::operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > **std::__debug::operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

5.37.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/bitset.

5.38 bitset File Reference

Classes

- class std::__profile::bitset< _Nb >

  Class std::bitset wrapper with performance instrumentation.
- struct std::hash< __profile::bitset< _Nb > >

  std::hash specialization for bitset.

Namespaces

- namespace std
- namespace std::__profile

Functions

- template<size_t _Nb>
  bitset< _Nb > **std::__profile::operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_ostream< _CharT,
  _Traits > & **std::__profile::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _CharT , typename _Traits , size_t _Nb>
  std::basic_istream< _CharT,
  _Traits > & **std::__profile::operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  bitset< _Nb > **std::__profile::operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  bitset< _Nb > **std::__profile::operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

5.38.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/bitset.

5.39 bool_set File Reference

Classes

- class std::tr2::bool_set

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_BOOL_SET**

Functions

- bool **std::tr2::certainly** (bool_set __b)
- bool **std::tr2::contains** (bool_set __s, bool_set __t)
- bool **std::tr2::equals** (bool_set __s, bool_set __t)
- bool **std::tr2::is_emptyset** (bool_set __b)
- bool **std::tr2::is_indeterminate** (bool_set __b)
- bool **std::tr2::is_singleton** (bool_set __b)

- bool_set **std::tr2::operator!=** (bool __s, bool_set __t)
- bool_set **std::tr2::operator!=** (bool_set __s, bool __t)
- bool_set **std::tr2::operator!=** (bool_set __s, bool_set __t)
- bool_set **std::tr2::operator&** (bool __s, bool_set __t)
- bool_set **std::tr2::operator&** (bool_set __s, bool __t)
- bool_set **std::tr2::operator==** (bool __s, bool_set __t)
- bool_set **std::tr2::operator==** (bool_set __s, bool __t)
- bool_set **std::tr2::operator^** (bool __s, bool_set __t)
- bool_set **std::tr2::operator^** (bool_set __s, bool __t)
- bool_set **std::tr2::operator|** (bool __s, bool_set __t)
- bool_set **std::tr2::operator|** (bool_set __s, bool __t)
- bool **std::tr2::possibly** (bool_set __b)
- bool_set **std::tr2::set_complement** (bool_set __b)
- bool_set **std::tr2::set_intersection** (bool __s, bool_set __t)
- bool_set **std::tr2::set_intersection** (bool_set __s, bool __t)
- bool_set **std::tr2::set_intersection** (bool_set __s, bool_set __t)
- bool_set **std::tr2::set_union** (bool __s, bool_set __t)
- bool_set **std::tr2::set_union** (bool_set __s, bool __t)
- bool_set **std::tr2::set_union** (bool_set __s, bool_set __t)

5.39.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file bool_set.

5.40 bool_set.tcc File Reference

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_BOOL_SET_TCC**

5.40.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file bool_set.tcc.

5.41 boost_concept_check.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_GLIBCXX_CLASS_REQUIRES**(_type_var, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES2**(_type_var1, _type_var2, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES3**(_type_var1, _type_var2, _type_var3, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES4**(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)
- #define **_GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT**(_OP, _NAME)
- #define **_GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT**(_OP, _NAME)
- #define **_IsUnused**

Functions

- template<class _Tp >
  void **__gnu_cxx::__aux_require_boolean_expr** (const _Tp &__t)
- void **__gnu_cxx::__error_type_must_be_a_signed_integer_type** ()
- void **__gnu_cxx::__error_type_must_be_an_integer_type** ()
- void **__gnu_cxx::__error_type_must_be_an_unsigned_integer_type** ()
- template<class _Concept >
  void **__gnu_cxx::__function_requires** ()

5.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

Definition in file boost_concept_check.h.

5.42 branch_policy.hpp File Reference

Classes

- struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >

  Primary template, base class for branch structure policies.
- struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >

  Specialization for const iterators.

Namespaces

- namespace __gnu_pbds

5.42.1 Detailed Description

Contains a base class for branch policies.

Definition in file branch_policy.hpp.

5.43 c++0x_warning.h File Reference

5.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

Definition in file c++0x_warning.h.

5.44 c++allocator.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp >
  using std::__allocator_base = __gnu_cxx::new_allocator< _Tp >

5.44.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file c++allocator.h.

5.45 c++config.h File Reference

Namespaces

- namespace std

Macros

- #define **__GLIBCXX__**
- #define **__glibcxx_assert**(_Condition)
- #define **__N**(msgid)
- #define **_GLIBCXX_ATOMIC_BUILTINS**
- #define **_GLIBCXX_BEGIN_EXTERN_C**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL**
- #define **_GLIBCXX_BEGIN_NAMESPACE_VERSION**
- #define **_GLIBCXX_DEPRECATED**
- #define **_GLIBCXX_END_EXTERN_C**
- #define **_GLIBCXX_END_NAMESPACE_LDBL**
- #define **_GLIBCXX_END_NAMESPACE_VERSION**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_FAST_MATH**
- #define **_GLIBCXX_FULLY_DYNAMIC_STRING**
- #define **_GLIBCXX_HAS_GTHREADS**

- #define **_GLIBCXX_HAVE_ACOSF**
- #define **_GLIBCXX_HAVE_ACOSL**
- #define **_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE**
- #define **_GLIBCXX_HAVE_ASINF**
- #define **_GLIBCXX_HAVE_ASINL**
- #define **_GLIBCXX_HAVE_AT_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_ATAN2F**
- #define **_GLIBCXX_HAVE_ATAN2L**
- #define **_GLIBCXX_HAVE_ATANF**
- #define **_GLIBCXX_HAVE_ATANL**
- #define **_GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY**
- #define **_GLIBCXX_HAVE_CEILF**
- #define **_GLIBCXX_HAVE_CEILL**
- #define **_GLIBCXX_HAVE_COMPLEX_H**
- #define **_GLIBCXX_HAVE_COSF**
- #define **_GLIBCXX_HAVE_COSHF**
- #define **_GLIBCXX_HAVE_COSHL**
- #define **_GLIBCXX_HAVE_COSL**
- #define **_GLIBCXX_HAVE_DLFCN_H**
- #define **_GLIBCXX_HAVE_EBADMSG**
- #define **_GLIBCXX_HAVE_ECANCELED**
- #define **_GLIBCXX_HAVE_ECHILD**
- #define **_GLIBCXX_HAVE_EIDRM**
- #define **_GLIBCXX_HAVE_ENDIAN_H**
- #define **_GLIBCXX_HAVE_ENODATA**
- #define **_GLIBCXX_HAVE_ENOLINK**
- #define **_GLIBCXX_HAVE_ENOSPC**
- #define **_GLIBCXX_HAVE_ENOSR**
- #define **_GLIBCXX_HAVE_ENOSTR**
- #define **_GLIBCXX_HAVE_ENOTRECOVERABLE**
- #define **_GLIBCXX_HAVE_ENOTSUP**
- #define **_GLIBCXX_HAVE_EOVERFLOW**
- #define **_GLIBCXX_HAVE_EOWNERDEAD**
- #define **_GLIBCXX_HAVE_EPERM**
- #define **_GLIBCXX_HAVE_EPROTO**
- #define **_GLIBCXX_HAVE_ETIME**
- #define **_GLIBCXX_HAVE_ETIMEDOUT**
- #define **_GLIBCXX_HAVE_ETXTBSY**
- #define **_GLIBCXX_HAVE_EWOULDBLOCK**
- #define **_GLIBCXX_HAVE_EXECINFO_H**
- #define **_GLIBCXX_HAVE_EXPF**
- #define **_GLIBCXX_HAVE_EXPL**
- #define **_GLIBCXX_HAVE_FABSF**
- #define **_GLIBCXX_HAVE_FABSL**
- #define **_GLIBCXX_HAVE_FENV_H**
- #define **_GLIBCXX_HAVE_FINITE**
- #define **_GLIBCXX_HAVE_FINITEF**
- #define **_GLIBCXX_HAVE_FINITEL**
- #define **_GLIBCXX_HAVE_FLOAT_H**
- #define **_GLIBCXX_HAVE_FLOORF**
- #define **_GLIBCXX_HAVE_FLOORL**

- #define **_GLIBCXX_HAVE_FMODF**
- #define **_GLIBCXX_HAVE_FMODL**
- #define **_GLIBCXX_HAVE_FREXPF**
- #define **_GLIBCXX_HAVE_FREXPL**
- #define **_GLIBCXX_HAVE_GETIPINFO**
- #define **_GLIBCXX_HAVE_GETS**
- #define **_GLIBCXX_HAVE_HYPOT**
- #define **_GLIBCXX_HAVE_HYPOTF**
- #define **_GLIBCXX_HAVE_HYPOTL**
- #define **_GLIBCXX_HAVE_ICONV**
- #define **_GLIBCXX_HAVE_INT64_T**
- #define **_GLIBCXX_HAVE_INT64_T_LONG**
- #define **_GLIBCXX_HAVE_INTTYPES_H**
- #define **_GLIBCXX_HAVE_ISINF**
- #define **_GLIBCXX_HAVE_ISINFF**
- #define **_GLIBCXX_HAVE_ISINFL**
- #define **_GLIBCXX_HAVE_ISNAN**
- #define **_GLIBCXX_HAVE_ISNANF**
- #define **_GLIBCXX_HAVE_ISNANL**
- #define **_GLIBCXX_HAVE_ISWBLANK**
- #define **_GLIBCXX_HAVE_LC_MESSAGES**
- #define **_GLIBCXX_HAVE_LDEXPF**
- #define **_GLIBCXX_HAVE_LDEXPL**
- #define **_GLIBCXX_HAVE_LIBINTL_H**
- #define **_GLIBCXX_HAVE_LIMIT_AS**
- #define **_GLIBCXX_HAVE_LIMIT_DATA**
- #define **_GLIBCXX_HAVE_LIMIT_FSIZE**
- #define **_GLIBCXX_HAVE_LIMIT_RSS**
- #define **_GLIBCXX_HAVE_LIMIT_VMEM**
- #define **_GLIBCXX_HAVE_LINUX_FUTEX**
- #define **_GLIBCXX_HAVE_LOCALE_H**
- #define **_GLIBCXX_HAVE_LOG10F**
- #define **_GLIBCXX_HAVE_LOG10L**
- #define **_GLIBCXX_HAVE_LOGF**
- #define **_GLIBCXX_HAVE_LOGL**
- #define **_GLIBCXX_HAVE_MBSTATE_T**
- #define **_GLIBCXX_HAVE_MEMORY_H**
- #define **_GLIBCXX_HAVE_MODF**
- #define **_GLIBCXX_HAVE_MODFF**
- #define **_GLIBCXX_HAVE_MODFL**
- #define **_GLIBCXX_HAVE_POLL**
- #define **_GLIBCXX_HAVE_POWF**
- #define **_GLIBCXX_HAVE_POWL**
- #define **_GLIBCXX_HAVE_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_S_ISREG**
- #define **_GLIBCXX_HAVE_SETENV**
- #define **_GLIBCXX_HAVE_SINCOS**
- #define **_GLIBCXX_HAVE_SINCOSF**
- #define **_GLIBCXX_HAVE_SINCOSL**
- #define **_GLIBCXX_HAVE_SINF**
- #define **_GLIBCXX_HAVE_SINHF**

- #define **_GLIBCXX_HAVE_SINHL**
- #define **_GLIBCXX_HAVE_SINL**
- #define **_GLIBCXX_HAVE_SLEEP**
- #define **_GLIBCXX_HAVE_SQRTF**
- #define **_GLIBCXX_HAVE_SQRTL**
- #define **_GLIBCXX_HAVE_STDALIGN_H**
- #define **_GLIBCXX_HAVE_STDBOOL_H**
- #define **_GLIBCXX_HAVE_STDINT_H**
- #define **_GLIBCXX_HAVE_STDLIB_H**
- #define **_GLIBCXX_HAVE_STRERROR_L**
- #define **_GLIBCXX_HAVE_STRERROR_R**
- #define **_GLIBCXX_HAVE_STRING_H**
- #define **_GLIBCXX_HAVE_STRINGS_H**
- #define **_GLIBCXX_HAVE_STRTOF**
- #define **_GLIBCXX_HAVE_STRTOLD**
- #define **_GLIBCXX_HAVE_STRXFRM_L**
- #define **_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT**
- #define **_GLIBCXX_HAVE_SYS_IOCTL_H**
- #define **_GLIBCXX_HAVE_SYS_IPC_H**
- #define **_GLIBCXX_HAVE_SYS_PARAM_H**
- #define **_GLIBCXX_HAVE_SYS_RESOURCE_H**
- #define **_GLIBCXX_HAVE_SYS_SEM_H**
- #define **_GLIBCXX_HAVE_SYS_STAT_H**
- #define **_GLIBCXX_HAVE_SYS_SYSINFO_H**
- #define **_GLIBCXX_HAVE_SYS_TIME_H**
- #define **_GLIBCXX_HAVE_SYS_TYPES_H**
- #define **_GLIBCXX_HAVE_SYS_UIO_H**
- #define **_GLIBCXX_HAVE_TANF**
- #define **_GLIBCXX_HAVE_TANHF**
- #define **_GLIBCXX_HAVE_TANHL**
- #define **_GLIBCXX_HAVE_TANL**
- #define **_GLIBCXX_HAVE_TGMATH_H**
- #define **_GLIBCXX_HAVE_TLS**
- #define **_GLIBCXX_HAVE_UNISTD_H**
- #define **_GLIBCXX_HAVE_USLEEP**
- #define **_GLIBCXX_HAVE_VFWSCANF**
- #define **_GLIBCXX_HAVE_VSWSCANF**
- #define **_GLIBCXX_HAVE_VWSCANF**
- #define **_GLIBCXX_HAVE_WCHAR_H**
- #define **_GLIBCXX_HAVE_WCSTOF**
- #define **_GLIBCXX_HAVE_WCTYPE_H**
- #define **_GLIBCXX_HAVE_WRITEV**
- #define **_GLIBCXX_HOSTED**
- #define **_GLIBCXX_ICONV_CONST**
- #define **_GLIBCXX_INLINE_VERSION**
- #define **_GLIBCXX_NAMESPACE_LDBL**
- #define **_GLIBCXX_PACKAGE__GLIBCXX_VERSION**
- #define **_GLIBCXX_PACKAGE_BUGREPORT**
- #define **_GLIBCXX_PACKAGE_NAME**
- #define **_GLIBCXX_PACKAGE_STRING**
- #define **_GLIBCXX_PACKAGE_TARNAME**

- #define **_GLIBCXX_PACKAGE_URL**
- #define **_GLIBCXX_PSEUDO_VISIBILITY**(V)
- #define **_GLIBCXX_RES_LIMITS**
- #define **_GLIBCXX_STDIO_EOF**
- #define **_GLIBCXX_STDIO_SEEK_CUR**
- #define **_GLIBCXX_STDIO_SEEK_END**
- #define **_GLIBCXX_SYMVER**
- #define **_GLIBCXX_SYMVER_GNU**
- #define **_GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER**(A)
- #define **_GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE**(A)
- #define **_GLIBCXX_THROW_OR_ABORT**(_EXC)
- #define **_GLIBCXX_USE_C99**
- #define **_GLIBCXX_USE_C99_COMPLEX**
- #define **_GLIBCXX_USE_C99_COMPLEX_TR1**
- #define **_GLIBCXX_USE_C99_CTYPE_TR1**
- #define **_GLIBCXX_USE_C99_FENV_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1**
- #define **_GLIBCXX_USE_C99_MATH**
- #define **_GLIBCXX_USE_C99_MATH_TR1**
- #define **_GLIBCXX_USE_C99_STDINT_TR1**
- #define **_GLIBCXX_USE_DECIMAL_FLOAT**
- #define **_GLIBCXX_USE_DEPRECATED**
- #define **_GLIBCXX_USE_FLOAT128**
- #define **_GLIBCXX_USE_GET_NPROCS**
- #define **_GLIBCXX_USE_GETTIMEOFDAY**
- #define **_GLIBCXX_USE_INT128**
- #define **_GLIBCXX_USE_LFS**
- #define **_GLIBCXX_USE_LONG_LONG**
- #define **_GLIBCXX_USE_NLS**
- #define **_GLIBCXX_USE_RANDOM_TR1**
- #define **_GLIBCXX_USE_SC_NPROCESSORS_ONLN**
- #define **_GLIBCXX_USE_WCHAR_T**
- #define **_GLIBCXX_VERBOSE**
- #define **_GLIBCXX_VISIBILITY**(V)
- #define **_GLIBCXX_WEAK_DEFINITION**
- #define **_GLIBCXX_X86_RDRAND**
- #define **_GTHREAD_USE_MUTEX_TIMEDLOCK**
- #define **LT_OBJDIR**
- #define **STDC_HEADERS**

Typedefs

- typedef __PTRDIFF_TYPE__ **std::ptrdiff_t**
- typedef __SIZE_TYPE__ **std::size_t**

Variables

- decltype(nullptr) typedef **std::nullptr_t**

5.45.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

Definition in file c++config.h.

5.46 c++io.h File Reference

Namespaces

- namespace std

Typedefs

- typedef FILE **std::__c_file**
- typedef __gthread_mutex_t **std::__c_lock**

5.46.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

Definition in file c++io.h.

5.47 c++locale.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_C_LOCALE_GNU**
- #define **_GLIBCXX_NUM_CATEGORIES**

Typedefs

- typedef __locale_t **std::__c_locale**

Functions

- int **std::__convert_from_v** (const __c_locale &__cloc __attribute__((__unused__)), char ∗__out, const int __size __attribute__((__unused__)), const char ∗__fmt,...)

5.47.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file c++locale.h.

5.48 c++locale_internal.h File Reference

5.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file c++locale_internal.h.

5.49 cassert File Reference

5.49.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cassert.

5.50 cast.h File Reference

Classes

- struct __gnu_cxx::_Caster< _ToType >

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _ToType , typename _FromType >
  _ToType **__gnu_cxx::__const_pointer_cast** (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__gnu_cxx::__const_pointer_cast** (_FromType *__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__gnu_cxx::__dynamic_pointer_cast** (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__gnu_cxx::__dynamic_pointer_cast** (_FromType *__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__gnu_cxx::__reinterpret_pointer_cast** (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType **__gnu_cxx::__reinterpret_pointer_cast** (_FromType *__arg)
- template<typename _ToType , typename _FromType >
  _ToType __gnu_cxx::__static_pointer_cast (const _FromType &__arg)
- template<typename _ToType , typename _FromType >
  _ToType __gnu_cxx::__static_pointer_cast (_FromType *__arg)

5.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/pointer.h>.

Definition in file cast.h.

5.51 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference

5.51.1 Detailed Description

Contains a resize trigger implementation.

Definition in file cc_hash_max_collision_check_resize_trigger_imp.hpp.

5.52 cc_ht_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CC_HASH_NAME**
- #define **PB_DS_CC_HASH_TRAITS_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_HASH_EQ_FN_C_DEC**
- #define **PB_DS_RANGED_HASH_FN_C_DEC**

5.52.1 Detailed Description

Contains an implementation class for cc_ht_map_.

Definition in file cc_ht_map_.hpp.

5.53 ccomplex File Reference

Macros

- #define **_GLIBCXX_CCOMPLEX**

5.53.1 Detailed Description

This is a Standard C++ Library header.

Definition in file ccomplex.

5.54 ccomplex File Reference

Macros

- #define **_GLIBCXX_TR1_CCOMPLEX**

5.54.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/ccomplex.

5.55 cctype File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CCTYPE**

5.55.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the ∗.h implementation files.

This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cctype.

5.56 cctype File Reference

Macros

- #define **_GLIBCXX_TR1_CCTYPE**

5.56.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cctype.

5.57 cerrno File Reference

Macros

- #define **_GLIBCXX_CERRNO**
- #define **errno**

5.57.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cerrno.

5.58 cfenv File Reference

Macros

- #define **_GLIBCXX_CFENV**

5.58.1 Detailed Description

This is a Standard C++ Library header.

Definition in file cfenv.

5.59 cfenv File Reference

Macros

- #define **_GLIBCXX_TR1_CFENV**

5.59.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cfenv.

5.60 cfloat File Reference

Macros

- #define **_GLIBCXX_CFLOAT**
- #define **DECIMAL_DIG**
- #define **FLT_EVAL_METHOD**

5.60.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cfloat.

5.61 cfloat File Reference

Macros

- #define **_GLIBCXX_TR1_CFLOAT**

5.61.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cfloat.

5.62 char_traits.h File Reference

Classes

- struct __gnu_cxx::_Char_types< _CharT >

  Mapping from character type to associated types.
- struct __gnu_cxx::char_traits< _CharT >

  Base class used to implement std::char_traits.
- struct std::char_traits< _CharT >

  Basis for explicit traits specializations.
- struct std::char_traits< char >

  21.1.3.1 char_traits specializations
- struct std::char_traits< wchar_t >

  21.1.3.2 char_traits specializations

Namespaces

- namespace __gnu_cxx
- namespace std

5.62.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

Definition in file char_traits.h.

5.63 checkers.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter , typename _Compare >
  bool __gnu_parallel::__is_sorted (_IIter __begin, _IIter __end, _Compare __comp)

5.63.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file checkers.h.

5.64 chrono File Reference

Classes

- struct std::chrono::duration< _Rep, _Period >

  duration
- struct std::chrono::duration_values< _Rep >

  duration_values
- struct std::chrono::system_clock

  system_clock
- struct std::chrono::time_point< _Clock, _Dur >

  time_point
- struct std::chrono::treat_as_floating_point< _Rep >

  treat_as_floating_point

Namespaces

- namespace std
- namespace std::chrono

Macros

- #define **_GLIBCXX_CHRONO**

Typedefs

- typedef system_clock **std::chrono::high_resolution_clock**
- typedef duration< int, ratio< 3600 > > std::chrono::hours
- typedef duration< int64_t, micro > std::chrono::microseconds
- typedef duration< int64_t, milli > std::chrono::milliseconds
- typedef duration< int, ratio< 60 > > std::chrono::minutes

- typedef duration< int64_t, nano > std::chrono::nanoseconds
- typedef duration< int64_t > std::chrono::seconds
- typedef system_clock **std::chrono::steady_clock**

Functions

- template<typename _ToDur , typename _Rep , typename _Period >
  constexpr enable_if
  < __is_duration< _ToDur >
  ::value, _ToDur >::type std::chrono::duration_cast (const duration< _Rep, _Period > &__d)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **std::chrono::operator!=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **std::chrono::operator!=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period , typename _Rep2 >
  constexpr duration< typename
  __common_rep_type< _Rep1,
  typename enable_if
  <!__is_duration< _Rep2 >
  ::value, _Rep2 >::type >::type,
  _Period > **std::chrono::operator%** (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type
  < duration< _Rep1, _Period1 >
  , duration< _Rep2, _Period2 >
  >::type **std::chrono::operator%** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1 , typename _Period , typename _Rep2 >
  constexpr duration< typename
  __common_rep_type< _Rep1,
  _Rep2 >::type, _Period > **std::chrono::operator∗** (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1 , typename _Rep2 , typename _Period >
  constexpr duration< typename
  __common_rep_type< _Rep2,
  _Rep1 >::type, _Period > **std::chrono::operator∗** (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type
  < duration< _Rep1, _Period1 >
  , duration< _Rep2, _Period2 >
  >::type **std::chrono::operator+** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Rep2 , typename _Period2 >
  constexpr time_point< _Clock,
  typename common_type< _Dur1,
  duration< _Rep2, _Period2 >
  >::type > **std::chrono::operator+** (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1 , typename _Period1 , typename _Clock , typename _Dur2 >
  constexpr time_point< _Clock,
  typename common_type< duration
  < _Rep1, _Period1 >, _Dur2 >
  ::type > **std::chrono::operator+** (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type
  < duration< _Rep1, _Period1 >
  , duration< _Rep2, _Period2 >
  >::type **std::chrono::operator-** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Rep2 , typename _Period2 >
  constexpr time_point< _Clock,
  typename common_type< _Dur1,
  duration< _Rep2, _Period2 >
  >::type > **std::chrono::operator-** (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr common_type< _Dur1,
  _Dur2 >::type **std::chrono::operator-** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period , typename _Rep2 >
  constexpr duration< typename
  __common_rep_type< _Rep1,
  typename enable_if
  <!__is_duration< _Rep2 >
  ::value, _Rep2 >::type >::type,
  _Period > **std::chrono::operator/** (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr common_type< _Rep1,
  _Rep2 >::type **std::chrono::operator/** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **std::chrono::operator<** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **std::chrono::operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **std::chrono::operator<=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **std::chrono::operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **std::chrono::operator==** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **std::chrono::operator==** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **std::chrono::operator>** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **std::chrono::operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
  constexpr bool **std::chrono::operator>=** (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock , typename _Dur1 , typename _Dur2 >
  constexpr bool **std::chrono::operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _ToDur , typename _Clock , typename _Dur >
  constexpr enable_if
  < __is_duration< _ToDur >
  ::value, time_point< _Clock,
  _ToDur > >::type std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)

5.64.1 Detailed Description

This is a Standard C++ Library header.

Definition in file chrono.

5.65 cinttypes File Reference

Macros

- #define **_GLIBCXX_CINTTYPES**

5.65.1 Detailed Description

This is a Standard C++ Library header.

Definition in file cinttypes.

5.66 cinttypes File Reference

Macros

- #define **_GLIBCXX_TR1_CINTTYPES**

5.66.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cinttypes.

5.67 ciso646 File Reference

5.67.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file ciso646.

5.68 climits File Reference

Macros

- #define **_GLIBCXX_CLIMITS**
- #define **LLONG_MAX**
- #define **LLONG_MIN**
- #define **ULLONG_MAX**

5.68.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file climits.

5.69 climits File Reference

Macros

- #define **_GLIBCXX_TR1_CLIMITS**

5.69.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/climits.

5.70 clocale File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CLOCALE**

5.70.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file clocale.

5.71 cmath File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CMATH**

Functions

- constexpr double **std::abs** (double __x)
- constexpr float **std::abs** (float __x)
- constexpr long double **std::abs** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::abs** (_Tp __x)
- constexpr float **std::acos** (float __x)
- constexpr long double **std::acos** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::acos** (_Tp __x)
- constexpr float **std::asin** (float __x)
- constexpr long double **std::asin** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::asin** (_Tp __x)
- constexpr float **std::atan** (float __x)
- constexpr long double **std::atan** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::atan** (_Tp __x)
- constexpr float **std::atan2** (float __y, float __x)
- constexpr long double **std::atan2** (long double __y, long double __x)
- template<typename _Tp , typename _Up >
  constexpr
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **std::atan2** (_Tp __y, _Up __x)

- constexpr float **std::ceil** (float __x)
- constexpr long double **std::ceil** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::ceil** (_Tp __x)
- constexpr float **std::cos** (float __x)
- constexpr long double **std::cos** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::cos** (_Tp __x)
- constexpr float **std::cosh** (float __x)
- constexpr long double **std::cosh** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::cosh** (_Tp __x)
- constexpr float **std::exp** (float __x)
- constexpr long double **std::exp** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::exp** (_Tp __x)
- constexpr float **std::fabs** (float __x)
- constexpr long double **std::fabs** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::fabs** (_Tp __x)
- constexpr float **std::floor** (float __x)
- constexpr long double **std::floor** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::floor** (_Tp __x)
- constexpr float **std::fmod** (float __x, float __y)
- constexpr long double **std::fmod** (long double __x, long double __y)
- template<typename _Tp , typename _Up >
  constexpr
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **std::fmod** (_Tp __x, _Up __y)
- float **std::frexp** (float __x, int ∗__exp)
- long double **std::frexp** (long double __x, int ∗__exp)

- template< typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::frexp** (_Tp __x, int ∗__exp)
- constexpr float **std::ldexp** (float __x, int __exp)
- constexpr long double **std::ldexp** (long double __x, int __exp)
- template< typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::ldexp** (_Tp __x, int __exp)
- constexpr float **std::log** (float __x)
- constexpr long double **std::log** (long double __x)
- template< typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::log** (_Tp __x)
- constexpr float **std::log10** (float __x)
- constexpr long double **std::log10** (long double __x)
- template< typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::log10** (_Tp __x)
- float **std::modf** (float __x, float ∗__iptr)
- long double **std::modf** (long double __x, long double ∗__iptr)
- constexpr float **std::pow** (float __x, float __y)
- constexpr long double **std::pow** (long double __x, long double __y)
- template< typename _Tp , typename _Up >
  constexpr
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **std::pow** (_Tp __x, _Up __y)
- constexpr float **std::sin** (float __x)
- constexpr long double **std::sin** (long double __x)
- template< typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::sin** (_Tp __x)
- constexpr float **std::sinh** (float __x)
- constexpr long double **std::sinh** (long double __x)
- template< typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::sinh** (_Tp __x)
- constexpr float **std::sqrt** (float __x)
- constexpr long double **std::sqrt** (long double __x)

- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::sqrt** (_Tp __x)
- constexpr float **std::tan** (float __x)
- constexpr long double **std::tan** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::tan** (_Tp __x)
- constexpr float **std::tanh** (float __x)
- constexpr long double **std::tanh** (long double __x)
- template<typename _Tp >
  constexpr
  __gnu_cxx::__enable_if
  < __is_integer< _Tp >::__value,
  double >::__type **std::tanh** (_Tp __x)

5.71.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cmath.

5.72 cmath File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CMATH**

Functions

- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)

- template<typename _Tpx , typename _Tpy >
  __gnu_cxx::__promote_2< _Tpx,
  _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp , typename _Tpn >
  __gnu_cxx::__promote_2< _Tp,
  _Tpn >::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa , typename _Tpc , typename _Tp >
  __gnu_cxx::__promote_3< _Tpa,
  _Tpc, _Tp >::__type std::tr1::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float **std::tr1::conf_hypergf** (float __a, float __c, float __x)
- long double **std::tr1::conf_hypergl** (long double __a, long double __c, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu , typename _Tp >
  __gnu_cxx::__promote_2< _Tpnu,
  _Tp >::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp , typename _Tpp >
  __gnu_cxx::__promote_2< _Tp,
  _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp , typename _Tpp >
  __gnu_cxx::__promote_2< _Tp,
  _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)

- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp , typename _Tpn , typename _Tpp >
  __gnu_cxx::__promote_3< _Tp,
  _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
  __gnu_cxx::__promote_4< _Tpa,
  _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **std::tr1::hypergf** (float __a, float __b, float __c, float __x)
- long double **std::tr1::hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- double **std::tr1::pow** (double __x, double __y)
- float **std::tr1::pow** (float __x, float __y)
- long double **std::tr1::pow** (long double __x, long double __y)
- template<typename _Tp , typename _Up >
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type **std::tr1::pow** (_Tp __x, _Up __y)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

5.72.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cmath.

5.73 cmp_fn_imps.hpp File Reference

5.73.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container comparison related functions.

Definition in file cmp_fn_imps.hpp.

5.74 codecvt.h File Reference

Classes

- class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >

  Common base for codecvt functions.
- class std::codecvt< _InternT, _ExternT, _StateT >

  Primary class template codecvt.
  NB: Generic, mostly useless implementation.
- class std::codecvt< char, char, mbstate_t >

  class codecvt<char, char, mbstate_t> specialization.
- class std::codecvt< wchar_t, char, mbstate_t >

  class codecvt<wchar_t, char, mbstate_t> specialization.
- class std::codecvt_base

  Empty base class for codecvt facet [22.2.1.5].
- class std::codecvt_byname< _InternT, _ExternT, _StateT >

  class codecvt_byname [22.2.1.6].

Namespaces

- namespace std

5.74.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file codecvt.h.

5.75 codecvt_specializations.h File Reference

Classes

- struct __gnu_cxx::encoding_char_traits< _CharT >

  encoding_char_traits

- class __gnu_cxx::encoding_state

  Extension to use iconv for dealing with character encodings.
- class std::codecvt< _InternT, _ExternT, encoding_state >

  codecvt< InternT, _ExternT, encoding_state> specialization.

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Tp >
  size_t **std::__iconv_adaptor** (size_t(∗__func)(iconv_t, _Tp, size_t ∗, char ∗∗, size_t ∗), iconv_t __cd, char ∗∗__inbuf, size_t ∗__inbytes, char ∗∗__outbuf, size_t ∗__outbytes)

5.75.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file codecvt_specializations.h.

5.76 compatibility.h File Reference

5.76.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

Definition in file x86_64-unknown-linux-gnu/bits/compatibility.h.

5.77 compatibility.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _Tp >
  _Tp **__gnu_parallel::__add_omp** (volatile _Tp ∗__ptr, _Tp __addend)
- template<typename _Tp >
  bool **__gnu_parallel::__cas_omp** (volatile _Tp ∗__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp >
  bool __gnu_parallel::__compare_and_swap (volatile _Tp ∗__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp >
  _Tp __gnu_parallel::__fetch_and_add (volatile _Tp ∗__ptr, _Tp __addend)
- void __gnu_parallel::__yield ()

5.77.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations. This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

Definition in file parallel/compatibility.h.

5.78 compiletime_settings.h File Reference

Macros

- #define _GLIBCXX_ASSERTIONS
- #define _GLIBCXX_CALL(__n)
- #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
- #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
- #define _GLIBCXX_SCALE_DOWN_FPU
- #define _GLIBCXX_VERBOSE_LEVEL

5.78.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file compiletime_settings.h.

5.78.2 Macro Definition Documentation

5.78.2.1 #define _GLIBCXX_ASSERTIONS

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Should be switched on only locally.

Definition at line 61 of file compiletime_settings.h.

Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

5.78.2.2 #define _GLIBCXX_CALL(*__n*)

Macro to produce log message when entering a function.

Parameters

__n	Input size.

See Also

_GLIBCXX_VERBOSE_LEVEL

Definition at line 44 of file compiletime_settings.h.

Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_parallel::__merge_advance(), __gnu_parallel::__parallel_nth_element(), __gnu_parallel::__parallel_partial_sum(), __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort_qs(), __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__parallel_unique_copy(), __gnu_parallel::__search_template(), __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge(), __gnu_-

parallel::multiway_merge_3_variant(), __gnu_parallel::multiway_merge_4_variant(), __gnu_parallel::multiway_merge_loser_tree(), __gnu_parallel::multiway_merge_loser_tree_sentinel(), __gnu_parallel::multiway_merge_loser_tree_unguarded(), __gnu_parallel::multiway_merge_sentinels(), __gnu_parallel::parallel_multiway_merge(), and __gnu_parallel::parallel_sort_mwms().

5.78.2.3 #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Consider the size of the L1 cache for gnu_parallel::__parallel_random_shuffle().

Definition at line 68 of file compiletime_settings.h.

5.78.2.4 #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Consider the size of the TLB for gnu_parallel::__parallel_random_shuffle().

Definition at line 74 of file compiletime_settings.h.

5.78.2.5 #define _GLIBCXX_SCALE_DOWN_FPU

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file compiletime_settings.h.

5.78.2.6 #define _GLIBCXX_VERBOSE_LEVEL

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file compiletime_settings.h.

5.79 complex File Reference

Classes

- struct std::complex< _Tp >
- struct std::complex< double >

  26.2.3 complex specializations complex<double> specialization
- struct std::complex< float >

  26.2.3 complex specializations complex<float> specialization
- struct std::complex< long double >

  26.2.3 complex specializations complex<long double> specialization

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_GLIBCXX_COMPLEX**

Functions

- template<typename _Tp >
  _Tp **std::__complex_abs** (const complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_acos** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  _Tp **std::__complex_arg** (const complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_asin** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_asinh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_atan** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::__complex_proj** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > **std::__complex_tanh** (const complex< _Tp > &__z)
- template<typename _Tp >
  _Tp std::abs (const complex< _Tp > &)
- template<typename _Tp >
  std::complex< _Tp > std::acos (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::acosh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  _Tp std::arg (const complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)
- template<typename _Tp >
  std::complex< _Tp > std::asin (const std::complex< _Tp > &__z)

- template<typename _Tp >
  std::complex< _Tp > std::asinh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::atan (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::atanh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  complex< _Tp > std::conj (const complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **std::conj** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > std::cos (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::cosh (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::exp (const complex< _Tp > &)
- template<typename _Tp >
  _Tp std::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp >
  constexpr _Tp **std::imag** (const complex< _Tp > &__z)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **std::imag** (_Tp)
- template<typename _Tp >
  complex< _Tp > std::log (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::log10 (const complex< _Tp > &)
- template<typename _Tp >
  _Tp std::norm (const complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **std::norm** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > std::operator+ (const complex< _Tp > &__x)
- template<typename _Tp >
  complex< _Tp > std::operator- (const complex< _Tp > &__x)
- template<typename _Tp , typename _CharT , class _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- template<typename _Tp , typename _CharT , class _Traits >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template<typename _Tp >
  complex< _Tp > std::polar (const _Tp &, const _Tp &=0)
- template<typename _Tp >
  complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp >
  complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::pow** (const std::complex< _Tp > &__x, const _Up &__y)

- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::proj** (const std::complex< _Tp > &)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **std::proj** (_Tp __x)
- template<typename _Tp >
  constexpr _Tp **std::real** (const complex< _Tp > &__z)
- template<typename _Tp >
  __gnu_cxx::__promote< _Tp >::__type **std::real** (_Tp __x)
- template<typename _Tp >
  complex< _Tp > std::sin (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::sinh (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::sqrt (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::tan (const complex< _Tp > &)
- template<typename _Tp >
  complex< _Tp > std::tanh (const complex< _Tp > &)

- template<typename _Tp >
  complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > std::operator∗ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator∗ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  complex< _Tp > std::operator∗ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _Tp >
  complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp >
  constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp >
  constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)

5.79.1 Detailed Description

This is a Standard C++ Library header.

Definition in file complex.

5.80 complex File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_COMPLEX**

Functions

- template<typename _Tp >
  std::complex< _Tp > **std::tr1::__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::__complex_asinh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::acosh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::asinh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::atanh (const std::complex< _Tp > &__z)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::conj** (const std::complex< _Tp > &__z)

- template<typename _Tp >
  std::complex< typename
  __gnu_cxx::__promote< _Tp >
  ::__type > **std::tr1::conj** (_Tp __x)
- template<typename _Tp >
  std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::tr1::polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::tr1::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp , typename _Up >
  std::complex< typename
  __gnu_cxx::__promote_2< _Tp,
  _Up >::__type > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp >
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)

5.80.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/complex.

5.81 complex.h File Reference

Macros

- #define **_GLIBCXX_COMPLEX_H**

5.81.1 Detailed Description

This is a Standard C++ Library header.

Definition in file complex.h.

5.82 concept_check.h File Reference

Macros

- #define **__glibcxx_class_requires**(_a, _b)

- #define **__glibcxx_class_requires2**(_a, _b, _c)
- #define **__glibcxx_class_requires3**(_a, _b, _c, _d)
- #define **__glibcxx_class_requires4**(_a, _b, _c, _d, _e)
- #define **__glibcxx_function_requires**(...)

5.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

Definition in file concept_check.h.

5.83 concurrence.h File Reference

Classes

- class __gnu_cxx::__scoped_lock
  Scoped lock idiom.

Namespaces

- namespace __gnu_cxx

Enumerations

- enum **_Lock_policy** { **_S_single**, **_S_mutex**, **_S_atomic** }

Functions

- void **__gnu_cxx::__throw_concurrence_lock_error** ()
- void **__gnu_cxx::__throw_concurrence_unlock_error** ()

Variables

- static const _Lock_policy **__gnu_cxx::__default_lock_policy**

5.83.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file concurrence.h.

5.84 cond_dealtor.hpp File Reference

Classes

- class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
  Conditional deallocate constructor argument.

Namespaces

- namespace __gnu_pbds

5.84.1 Detailed Description

Contains a conditional deallocator.

Definition in file cond_dealtor.hpp.

5.85 cond_key_dtor_entry_dealtor.hpp File Reference

Classes

- class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

  Conditional deallocate constructor argument.

Namespaces

- namespace __gnu_pbds

5.85.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

Definition in file cond_key_dtor_entry_dealtor.hpp.

5.86 condition_variable File Reference

Classes

- class std::condition_variable

  condition_variable
- class std::condition_variable_any

  condition_variable_any

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CONDITION_VARIABLE**

Enumerations

- enum std::cv_status

5.86.1 Detailed Description

This is a Standard C++ Library header.

Definition in file condition_variable.

5.87 const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >

  Const point-type iterator.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BIN_HEAP_CIT_BASE**

5.87.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file binary_heap_/const_iterator.hpp.

5.88 const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >

  Const point-type iterator.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BASIC_HEAP_CIT_BASE**
- #define **PB_DS_CLASS_C_DEC**

5.88.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file left_child_next_sibling_heap_/const_iterator.hpp.

5.89 const_iterator.hpp File Reference

Classes

- class const_iterator_

  Const range-type iterator.

5.89.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.

Definition in file unordered_iterator/const_iterator.hpp.

5.90 constructor_destructor_fn_imps.hpp File Reference

5.90.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

Definition in file cc_hash_table_map_/constructor_destructor_fn_imps.hpp.

5.91 constructor_destructor_fn_imps.hpp File Reference

5.91.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

Definition in file gp_hash_table_map_/constructor_destructor_fn_imps.hpp.

5.92 constructor_destructor_fn_imps.hpp File Reference

5.93 constructor_destructor_no_store_hash_fn_imps.hpp File Reference

5.93.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

Definition in file cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp.

5.94 constructor_destructor_no_store_hash_fn_imps.hpp File Reference

5.94.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

Definition in file gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp.

5.95 constructor_destructor_store_hash_fn_imps.hpp File Reference

5.95.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

Definition in file cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp.

5.96 constructor_destructor_store_hash_fn_imps.hpp File Reference

5.96.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

Definition in file gp_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp.

5.97 constructors_destructor_fn_imps.hpp File Reference

5.97.1 Detailed Description

Contains an implementation class for binary_heap_.

Definition in file binary_heap_/constructors_destructor_fn_imps.hpp.

5.98 constructors_destructor_fn_imps.hpp File Reference

5.98.1 Detailed Description

Contains an implementation for binomial_heap_.

Definition in file binomial_heap_/constructors_destructor_fn_imps.hpp.

5.99 constructors_destructor_fn_imps.hpp File Reference

5.99.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_/constructors_destructor_fn_imps.hpp.

5.100 constructors_destructor_fn_imps.hpp File Reference

5.100.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/constructors_destructor_fn_imps.hpp.

5.101 constructors_destructor_fn_imps.hpp File Reference

5.101.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp.

5.102 constructors_destructor_fn_imps.hpp File Reference

5.102.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/constructors_destructor_fn_imps.hpp.

5.103 constructors_destructor_fn_imps.hpp File Reference

5.103.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_/constructors_destructor_fn_imps.hpp.

5.104 constructors_destructor_fn_imps.hpp File Reference

5.104.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/constructors_destructor_fn_imps.hpp.

5.105 constructors_destructor_fn_imps.hpp File Reference

5.105.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/constructors_destructor_fn_imps.hpp.

5.106 constructors_destructor_fn_imps.hpp File Reference

5.106.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

Definition in file rc_binomial_heap_/constructors_destructor_fn_imps.hpp.

5.107 constructors_destructor_fn_imps.hpp File Reference

5.107.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_tree_/constructors_destructor_fn_imps.hpp.

5.108 constructors_destructor_fn_imps.hpp File Reference

5.108.1 Detailed Description

Contains an implementation for thin_heap_.

Definition in file thin_heap_/constructors_destructor_fn_imps.hpp.

5.109 container_base_dispatch.hpp File Reference

Classes

- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >

  Specialization colision-chaining hash map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >

  Specialization general-probe hash map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >

  Specialization for list-update map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >

  Specialization ordered-vector tree map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >

  Specialization for PATRICIA trie map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >

  Specialization for R-B tree map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >

  Specialization splay tree map.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >

  Specialization colision-chaining hash set.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >

  Specialization general-probe hash set.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >

  Specialization for list-update set.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >

  Specialization ordered-vector tree set.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >

  Specialization for PATRICIA trie set.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >

  Specialization for R-B tree set.
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >

  Specialization splay tree set.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CHECK_KEY_DOES_NOT_EXIST**(_Key)
- #define **PB_DS_CHECK_KEY_EXISTS**(_Key)
- #define **PB_DS_DATA_FALSE_INDICATOR**
- #define **PB_DS_DATA_TRUE_INDICATOR**
- #define **PB_DS_DEBUG_VERIFY**(_Cond)
- #define **PB_DS_EP2VP**(X)
- #define **PB_DS_EP2VP**(X)
- #define **PB_DS_V2F**(X)
- #define **PB_DS_V2F**(X)
- #define **PB_DS_V2S**(X)
- #define **PB_DS_V2S**(X)

5.109.1 Detailed Description

Contains associative container dispatching.

Definition in file container_base_dispatch.hpp.

5.110 cpp_type_traits.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

5.110.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/type_traits>.

Definition in file cpp_type_traits.h.

5.111 cpu_defines.h File Reference

5.111.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

Definition in file cpu_defines.h.

5.112 csetjmp File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSETJMP**
- #define **setjmp**(env)

5.112.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file csetjmp.

5.113 csignal File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSIGNAL**

5.113.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file csignal.

5.114 cstdarg File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDARG**
- #define **va_end**(ap)

5.114.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cstdarg.

5.115 cstdarg File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDARG**

5.115.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cstdarg.

5.116 cstdbool File Reference

Macros

- #define **_GLIBCXX_CSTDBOOL**

5.116.1 Detailed Description

This is a Standard C++ Library header.

Definition in file cstdbool.

5.117 cstdbool File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDBOOL**

5.117.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cstdbool.

5.118 cstddef File Reference

5.118.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the $*$.h implementation files.

This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cstddef.

5.119 cstdint File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDINT**

5.119.1 Detailed Description

This is a Standard C++ Library header.

Definition in file cstdint.

5.120 cstdint File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CSTDINT**

5.120.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cstdint.

5.121 cstdio File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDIO**

Functions

- char ∗ **gets** (char ∗__s) __attribute__((deprecated))

5.121.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cstdio.

5.122 cstdio File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDIO**

5.122.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cstdio.

5.123 cstdlib File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDLIB**
- #define **EXIT_FAILURE**
- #define **EXIT_SUCCESS**

Functions

- void **std::abort** (void) throw ()
- int **std::atexit** (void(*)()) throw ()
- void **std::exit** (int) throw ()

5.123.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cstdlib.

5.124 cstdlib File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDLIB**

5.124.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cstdlib.

5.125 cstring File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTRING**

Functions

- void ∗ **std::memchr** (void ∗__s, int __c, size_t __n)
- char ∗ **std::strchr** (char ∗__s, int __n)
- char ∗ **std::strpbrk** (char ∗__s1, const char ∗__s2)
- char ∗ **std::strrchr** (char ∗__s, int __n)
- char ∗ **std::strstr** (char ∗__s1, const char ∗__s2)

5.125.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the ∗.h implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cstring.

5.126 ctgmath File Reference

Macros

- #define **_GLIBCXX_CTGMATH**

5.126.1 Detailed Description

This is a Standard C++ Library header.

Definition in file ctgmath.

5.127 ctgmath File Reference

Macros

- #define **_GLIBCXX_TR1_CTGMATH**

5.127.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/ctgmath.

5.128 ctime File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CTIME**

5.128.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the ∗.h implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file ctime.

5.129 ctime File Reference

Macros

- #define **_GLIBCXX_TR1_CTIME**

5.129.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/ctime.

5.130 ctype_base.h File Reference

Classes

- struct std::ctype_base
  Base class for ctype.

Namespaces

- namespace std

5.130.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file ctype_base.h.

5.131 ctype_inline.h File Reference

Namespaces

- namespace std

5.131.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file ctype_inline.h.

5.132 cwchar File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CWCHAR**

Functions

- wchar_t ∗ **std::wcschr** (wchar_t ∗__p, wchar_t __c)
- wchar_t ∗ **std::wcspbrk** (wchar_t ∗__s1, const wchar_t ∗__s2)
- wchar_t ∗ **std::wcsrchr** (wchar_t ∗__p, wchar_t __c)
- wchar_t ∗ **std::wcsstr** (wchar_t ∗__s1, const wchar_t ∗__s2)
- wchar_t ∗ **std::wmemchr** (wchar_t ∗__p, wchar_t __c, size_t __n)

5.132.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the ∗.h implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cwchar.

5.133 cwchar File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CWCHAR**

5.133.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cwchar.

5.134 cwctype File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CWCTYPE**

5.134.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

Definition in file cwctype.

5.135 cwctype File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CWCTYPE**

5.135.1 Detailed Description

This is a TR1 C++ Library header.

Definition in file tr1/cwctype.

5.136 cxxabi.h File Reference

Classes

- class __gnu_cxx::recursive_init_error

  Exception thrown by __cxa_guard_acquire.
  6.7[stmt.dcl]/4: If control re-enters the declaration (recursively) while the object is being initialized, the behavior is undefined.

Namespaces

- namespace __gnu_cxx
- namespace abi

Typedefs

- typedef __cxa_cdtor_return_type(∗ **__cxxabiv1::__cxa_cdtor_type**)(void ∗)

Functions

- __cxa_dependent_exception ∗ **__cxxabiv1::__cxa_allocate_dependent_exception** () noexcept
- void ∗ **__cxxabiv1::__cxa_allocate_exception** (size_t) noexcept
- int **__cxxabiv1::__cxa_atexit** (void(∗)(void ∗), void ∗, void ∗) noexcept
- void **__cxxabiv1::__cxa_bad_cast** () __attribute__((__noreturn__))
- void **__cxxabiv1::__cxa_bad_typeid** () __attribute__((__noreturn__))
- void ∗ **__cxxabiv1::__cxa_begin_catch** (void ∗) noexcept
- std::type_info ∗ **__cxxabiv1::__cxa_current_exception_type** () noexcept __attribute__((__pure__))
- void **__cxxabiv1::__cxa_deleted_virtual** (void) __attribute__((__noreturn__))
- char ∗ **__cxxabiv1::__cxa_demangle** (const char ∗__mangled_name, char ∗__output_buffer, size_t ∗__length, int ∗__status)
- void **__cxxabiv1::__cxa_end_catch** ()
- int **__cxxabiv1::__cxa_finalize** (void ∗)
- void **__cxxabiv1::__cxa_free_dependent_exception** (__cxa_dependent_exception ∗) noexcept
- void **__cxxabiv1::__cxa_free_exception** (void ∗) noexcept
- void ∗ **__cxxabiv1::__cxa_get_exception_ptr** (void ∗) noexcept __attribute__((__pure__))
- __cxa_eh_globals ∗ **__cxxabiv1::__cxa_get_globals** () noexcept __attribute__((__const__))
- __cxa_eh_globals ∗ **__cxxabiv1::__cxa_get_globals_fast** () noexcept __attribute__((__const__))
- void **__cxxabiv1::__cxa_guard_abort** (__guard ∗) noexcept
- int **__cxxabiv1::__cxa_guard_acquire** (__guard ∗)
- void **__cxxabiv1::__cxa_guard_release** (__guard ∗) noexcept
- void **__cxxabiv1::__cxa_pure_virtual** (void) __attribute__((__noreturn__))
- void **__cxxabiv1::__cxa_rethrow** () __attribute__((__noreturn__))
- int **__cxxabiv1::__cxa_thread_atexit** (void(∗)(void ∗), void ∗, void ∗) noexcept
- void **__cxxabiv1::__cxa_throw** (void ∗, std::type_info ∗, void(∗)(void ∗)) __attribute__((__noreturn__))

- __cxa_vec_ctor_return_type **__cxxabiv1::__cxa_vec_cctor** (void ∗__dest_array, void ∗__src_array, size_t __element_count, size_t __element_size, __cxa_cdtor_return_type(∗__constructor)(void ∗, void ∗), __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_cleanup** (void ∗__array_address, size_t __element_count, size_t __s, __cxa_cdtor_type __destructor) noexcept
- __cxa_vec_ctor_return_type **__cxxabiv1::__cxa_vec_ctor** (void ∗__array_address, size_t __element_count, size_t __element_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_delete** (void ∗__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_delete2** (void ∗__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor, void(∗__dealloc)(void ∗))
- void **__cxxabiv1::__cxa_vec_delete3** (void ∗__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor, void(∗__dealloc)(void ∗, size_t))
- void **__cxxabiv1::__cxa_vec_dtor** (void ∗__array_address, size_t __element_count, size_t __element_size, __cxa_cdtor_type __destructor)
- void ∗ **__cxxabiv1::__cxa_vec_new** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)
- void ∗ **__cxxabiv1::__cxa_vec_new2** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void ∗(∗__alloc)(size_t), void(∗__dealloc)(void ∗))
- void ∗ **__cxxabiv1::__cxa_vec_new3** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void ∗(∗__alloc)(size_t), void(∗__dealloc)(void ∗, size_t))
- void ∗ **__cxxabiv1::__dynamic_cast** (const void ∗__src_ptr, const __class_type_info ∗__src_type, const __class_type_info ∗__dst_type, ptrdiff_t __src2dst)

5.136.1 Detailed Description

The header provides an interface to the C++ ABI.

Definition in file cxxabi.h.

5.137 cxxabi_forced.h File Reference

Classes

- class __cxxabiv1::__forced_unwind

  Thrown as part of forced unwinding.
  A magic placeholder class that can be caught by reference to recognize forced unwinding.

5.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cxxabi.h>.

Definition in file cxxabi_forced.h.

5.138 cxxabi_tweaks.h File Reference

Macros

- #define **_GLIBCXX_CXA_VEC_CTOR_RETURN**(x)

- #define **_GLIBCXX_GUARD_BIT**
- #define **_GLIBCXX_GUARD_PENDING_BIT**
- #define **_GLIBCXX_GUARD_SET**(x)
- #define **_GLIBCXX_GUARD_TEST**(x)
- #define **_GLIBCXX_GUARD_WAITING_BIT**

Typedefs

- typedef void **__cxxabiv1::__cxa_cdtor_return_type**
- typedef void **__cxxabiv1::__cxa_vec_ctor_return_type**

Functions

- __extension__ typedef int __guard **__cxxabiv1::__attribute__** ((mode(__DI__)))

5.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cxxabi.h>.

Definition in file cxxabi_tweaks.h.

5.139 debug.h File Reference

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **__glibcxx_requires_cond**(_Cond, _Msg)
- #define **__glibcxx_requires_heap**(_First, _Last)
- #define **__glibcxx_requires_heap_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_non_empty_range**(_First, _Last)
- #define **__glibcxx_requires_nonempty**()
- #define **__glibcxx_requires_partitioned_lower**(_First, _Last, _Value)
- #define **__glibcxx_requires_partitioned_lower_pred**(_First, _Last, _Value, _Pred)
- #define **__glibcxx_requires_partitioned_upper**(_First, _Last, _Value)
- #define **__glibcxx_requires_partitioned_upper_pred**(_First, _Last, _Value, _Pred)
- #define **__glibcxx_requires_sorted**(_First, _Last)
- #define **__glibcxx_requires_sorted_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_sorted_set**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_sorted_set_pred**(_First1, _Last1, _First2, _Pred)
- #define **__glibcxx_requires_string**(_String)
- #define **__glibcxx_requires_string_len**(_String, _Len)
- #define **__glibcxx_requires_subscript**(_N)
- #define **__glibcxx_requires_valid_range**(_First, _Last)

- #define **_GLIBCXX_DEBUG_ASSERT**(_Condition)
- #define **_GLIBCXX_DEBUG_ONLY**(_Statement)
- #define **_GLIBCXX_DEBUG_PEDASSERT**(_Condition)

5.139.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug.h.

5.140 debug_allocator.h File Reference

Classes

- class __gnu_cxx::debug_allocator< _Alloc >

    A meta-allocator with debugging bits, as per [20.4].
    This is precisely the allocator defined in the C++ Standard.

Namespaces

- namespace __gnu_cxx

5.140.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file debug_allocator.h.

5.141 debug_fn_imps.hpp File Reference

5.141.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/debug_fn_imps.hpp.

5.142 debug_fn_imps.hpp File Reference

5.142.1 Detailed Description

Contains an implementation for binomial_heap_.

Definition in file binomial_heap_/debug_fn_imps.hpp.

5.143 debug_fn_imps.hpp File Reference

5.143.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_/debug_fn_imps.hpp.

5.144 debug_fn_imps.hpp File Reference

5.144.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/debug_fn_imps.hpp.

5.145 debug_fn_imps.hpp File Reference

5.145.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

Definition in file cc_hash_table_map_/debug_fn_imps.hpp.

5.146 debug_fn_imps.hpp File Reference

5.146.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

Definition in file gp_hash_table_map_/debug_fn_imps.hpp.

5.147 debug_fn_imps.hpp File Reference

5.147.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/debug_fn_imps.hpp.

5.148 debug_fn_imps.hpp File Reference

5.148.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

Definition in file list_update_map_/debug_fn_imps.hpp.

5.149 debug_fn_imps.hpp File Reference

5.149.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/debug_fn_imps.hpp.

5.150 debug_fn_imps.hpp File Reference

5.150.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_/debug_fn_imps.hpp.

5.151 debug_fn_imps.hpp File Reference

5.151.1 Detailed Description

Contains an implementation class for pat_trie_.

Definition in file pat_trie_/debug_fn_imps.hpp.

5.152 debug_fn_imps.hpp File Reference

5.152.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/debug_fn_imps.hpp.

5.153 debug_fn_imps.hpp File Reference

5.153.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

Definition in file rc_binomial_heap_/debug_fn_imps.hpp.

5.154 debug_fn_imps.hpp File Reference

5.154.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_tree_/debug_fn_imps.hpp.

5.155 debug_fn_imps.hpp File Reference

5.155.1 Detailed Description

Contains an implementation for thin_heap_.

Definition in file thin_heap_/debug_fn_imps.hpp.

5.156 debug_map_base.hpp File Reference

5.156.1 Detailed Description

Contains a debug-mode base for all maps.

Definition in file debug_map_base.hpp.

5.157 debug_no_store_hash_fn_imps.hpp File Reference

5.157.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

Definition in file cc_hash_table_map_/debug_no_store_hash_fn_imps.hpp.

5.158 debug_no_store_hash_fn_imps.hpp File Reference

5.158.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

Definition in file gp_hash_table_map_/debug_no_store_hash_fn_imps.hpp.

5.159 debug_store_hash_fn_imps.hpp File Reference

5.159.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

Definition in file cc_hash_table_map_/debug_store_hash_fn_imps.hpp.

5.160 debug_store_hash_fn_imps.hpp File Reference

5.160.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

Definition in file gp_hash_table_map_/debug_store_hash_fn_imps.hpp.

5.161 decimal File Reference

Classes

- class std::decimal::decimal128

  3.2.4 Class decimal128.
- class std::decimal::decimal32

  3.2.2 Class decimal32.
- class std::decimal::decimal64

  3.2.3 Class decimal64.

Namespaces

- namespace std
- namespace std::decimal

Macros

- #define **_DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_DEC**(_Op, _T1, _T2, _T3)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_INT**(_Op, _Tp)
- #define **_DECLARE_DECIMAL_COMPARISON**(_Op, _Tp)
- #define **_GLIBCXX_DECIMAL**
- #define **_GLIBCXX_USE_DECIMAL_**

Functions

- double **std::decimal::decimal128_to_double** (decimal128 __d)
- float **std::decimal::decimal128_to_float** (decimal128 __d)
- long double **std::decimal::decimal128_to_long_double** (decimal128 __d)
- long long **std::decimal::decimal128_to_long_long** (decimal128 __d)
- double **std::decimal::decimal32_to_double** (decimal32 __d)
- float **std::decimal::decimal32_to_float** (decimal32 __d)
- long double **std::decimal::decimal32_to_long_double** (decimal32 __d)
- long long std::decimal::decimal32_to_long_long (decimal32 __d)
- double **std::decimal::decimal64_to_double** (decimal64 __d)
- float **std::decimal::decimal64_to_float** (decimal64 __d)
- long double **std::decimal::decimal64_to_long_double** (decimal64 __d)
- long long **std::decimal::decimal64_to_long_long** (decimal64 __d)
- double **std::decimal::decimal_to_double** (decimal32 __d)
- double **std::decimal::decimal_to_double** (decimal64 __d)
- double **std::decimal::decimal_to_double** (decimal128 __d)
- float **std::decimal::decimal_to_float** (decimal32 __d)
- float **std::decimal::decimal_to_float** (decimal64 __d)
- float **std::decimal::decimal_to_float** (decimal128 __d)
- long double **std::decimal::decimal_to_long_double** (decimal32 __d)
- long double **std::decimal::decimal_to_long_double** (decimal64 __d)
- long double **std::decimal::decimal_to_long_double** (decimal128 __d)
- long long **std::decimal::decimal_to_long_long** (decimal32 __d)
- long long **std::decimal::decimal_to_long_long** (decimal64 __d)
- long long **std::decimal::decimal_to_long_long** (decimal128 __d)
- static decimal128 **std::decimal::make_decimal128** (long long __coeff, int __exp)
- static decimal128 **std::decimal::make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (unsigned long long __coeff, int __exp)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long __rhs)

- bool **std::decimal::operator!=** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator**∗ (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator**∗ (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator**∗ (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator**∗ (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator**∗ (unsigned long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator**∗ (long long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator**∗ (unsigned long long __lhs, decimal32 __rhs)

- decimal64 **std::decimal::operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator*** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (long long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator+** (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (long long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal64 __rhs)

- decimal64 **std::decimal::operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator+** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator+** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (long long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator-** (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (long long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long __rhs)

- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator-** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator-** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (long long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator/** (int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (unsigned int __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (long long __lhs, decimal32 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal32 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator/** (int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (unsigned int __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (long __lhs, decimal64 __rhs)

- decimal64 **std::decimal::operator/** (unsigned long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (long long __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal32 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal64 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator/** (long long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (unsigned int __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator**< (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**< (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**< (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator**< (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**< (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**< (unsigned long __lhs, decimal128 __rhs)

- bool **std::decimal::operator**< (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**< (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**< (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**< (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal32 __rhs)

- bool **std::decimal::operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator**> (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator**> (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**> (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator**> (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**> (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**> (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator**> (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (long __lhs, decimal128 __rhs)

- bool **std::decimal::operator>** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal128 __rhs)

5.161.1 Detailed Description

This is a Standard C++ Library header.

Definition in file decimal.

5.162 deque File Reference

Macros

- #define **_GLIBCXX_DEQUE**

5.162.1 Detailed Description

This is a Standard C++ Library header.

Definition in file deque.

5.163 deque File Reference

Classes

- class std::__debug::deque< _Tp, _Allocator >

  Class std::deque with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_DEQUE**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator!=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator>** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator>=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__debug::swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)

5.163.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/deque.

5.164 deque File Reference

Classes

- class std::__profile::deque< _Tp, _Allocator >

  Class std::deque wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Macros

- #define **_GLIBCXX_PROFILE_DEQUE**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator!=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator**< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator**<**=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator**> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator**>**=** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__profile::swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs)

5.164.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/deque.

5.165 deque.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_DEQUE_TCC**

Functions

- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::copy** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::copy_backward** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  void **std::fill** (const _Deque_iterator< _Tp, _Tp &, _Tp ∗ > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp ∗ > &__last, const _Tp &__value)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::move** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::move_backward** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)

5.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <deque>.

Definition in file deque.tcc.

5.166 direct_mask_range_hashing_imp.hpp File Reference

5.166.1 Detailed Description

Contains a range-hashing policy implementation

Definition in file direct_mask_range_hashing_imp.hpp.

5.167 direct_mod_range_hashing_imp.hpp File Reference

5.167.1 Detailed Description

Contains a range-hashing policy implementation

Definition in file direct_mod_range_hashing_imp.hpp.

5.168 dynamic_bitset File Reference

Classes

- struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >

  The dynamic_bitset class represents a sequence of bits.
- class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

Namespaces

- namespace std
- namespace std::tr2
- namespace std::tr2::__detail

Macros

- #define **_GLIBCXX_TR2_DYNAMIC_BITSET**

Functions

- template<typename _WordT , typename _Alloc >
  bool std::tr2::operator== (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool std::tr2::operator< (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT , typename _Alloc >
  bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

- template<typename _WordT , typename _Alloc >
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

- template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
  std::basic_istream< _CharT,
  _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)
- template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
  std::basic_ostream< _CharT,
  _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)

5.168.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file dynamic_bitset.

5.169 enc_filebuf.h File Reference

Classes

- class __gnu_cxx::enc_filebuf< _CharT >

  class enc_filebuf.

Namespaces

- namespace __gnu_cxx

5.169.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file enc_filebuf.h.

5.170 entry_cmp.hpp File Reference

Classes

- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >

  Specialization, false.
- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type

  Compare plus entry.
- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >

  Specialization, true.
- struct entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >

  Entry compare, primary template.

Namespaces

- namespace __gnu_pbds

5.170.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file entry_cmp.hpp.

5.171 entry_list_fn_imps.hpp File Reference

5.171.1 Detailed Description

Contains implementations of cc_ht_map_'s entry-list related functions.

Definition in file entry_list_fn_imps.hpp.

5.172 entry_metadata_base.hpp File Reference

Namespaces

- namespace __gnu_pbds

5.172.1 Detailed Description

Contains an implementation for a list update map.

Definition in file entry_metadata_base.hpp.

5.173 entry_pred.hpp File Reference

Classes

- struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >

  Specialization, false.
- struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >

  Specialization, true.
- struct entry_pred< _VTp, Pred, _Alloc, No_Throw >

  Entry predicate primary class template.

Namespaces

- namespace __gnu_pbds

5.173.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file entry_pred.hpp.

5.174 eq_by_less.hpp File Reference

Classes

- struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >

  Equivalence function.

Namespaces

- namespace __gnu_pbds

5.174.1 Detailed Description

Contains an equivalence function.

Definition in file eq_by_less.hpp.

5.175 equally_split.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _DifferenceType , typename _OutputIterator >
  _OutputIterator __gnu_parallel::__equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)
- template<typename _DifferenceType >
  _DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)

5.175.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file equally_split.h.

5.176 erase_fn_imps.hpp File Reference

5.176.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/erase_fn_imps.hpp.

5.177 erase_fn_imps.hpp File Reference

5.177.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_/erase_fn_imps.hpp.

5.178 erase_fn_imps.hpp File Reference

5.178.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/erase_fn_imps.hpp.

5.179 erase_fn_imps.hpp File Reference

5.179.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions.

Definition in file cc_hash_table_map_/erase_fn_imps.hpp.

5.180 erase_fn_imps.hpp File Reference

5.180.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions.

Definition in file gp_hash_table_map_/erase_fn_imps.hpp.

5.181 erase_fn_imps.hpp File Reference

5.181.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/erase_fn_imps.hpp.

5.182 erase_fn_imps.hpp File Reference

5.182.1 Detailed Description

Contains implementations of lu_map_.

Definition in file list_update_map_/erase_fn_imps.hpp.

5.183 erase_fn_imps.hpp File Reference

5.183.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/erase_fn_imps.hpp.

5.184 erase_fn_imps.hpp File Reference

5.184.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_/erase_fn_imps.hpp.

5.185 erase_fn_imps.hpp File Reference

5.185.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/erase_fn_imps.hpp.

5.186 erase_fn_imps.hpp File Reference

5.186.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/erase_fn_imps.hpp.

5.187 erase_fn_imps.hpp File Reference

5.187.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

Definition in file rc_binomial_heap_/erase_fn_imps.hpp.

5.188 erase_fn_imps.hpp File Reference

5.188.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_tree_/erase_fn_imps.hpp.

5.189 erase_fn_imps.hpp File Reference

5.189.1 Detailed Description

Contains an implementation for thin_heap_.

Definition in file thin_heap_/erase_fn_imps.hpp.

5.190 erase_no_store_hash_fn_imps.hpp File Reference

5.190.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions, when the hash value is not stored.

Definition in file cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp.

5.191 erase_no_store_hash_fn_imps.hpp File Reference

5.191.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions, when the hash value is not stored.

Definition in file gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp.

5.192 erase_store_hash_fn_imps.hpp File Reference

5.192.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions, when the hash value is stored.

Definition in file cc_hash_table_map_/erase_store_hash_fn_imps.hpp.

5.193 erase_store_hash_fn_imps.hpp File Reference

5.193.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions, when the hash value is stored.

Definition in file gp_hash_table_map_/erase_store_hash_fn_imps.hpp.

5.194 error_constants.h File Reference

Namespaces

- namespace std

Enumerations

- enum **errc**

5.194.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <system_error>.

Definition in file error_constants.h.

5.195 exception File Reference

Classes

- class std::bad_exception
- class std::exception

  Base class for all library exceptions.

Namespaces

- namespace __gnu_cxx
- namespace std

Typedefs

- typedef void(∗ std::terminate_handler)()
- typedef void(∗ std::unexpected_handler)()

Functions

- void __gnu_cxx::__verbose_terminate_handler ()
- terminate_handler std::set_terminate (terminate_handler) noexcept
- unexpected_handler std::set_unexpected (unexpected_handler) noexcept
- void std::terminate () noexcept __attribute__((__noreturn__))
- bool std::uncaught_exception () noexcept __attribute__((__pure__))
- void std::unexpected () __attribute__((__noreturn__))

5.195.1 Detailed Description

This is a Standard C++ Library header.

Definition in file exception.

5.196 exception.hpp File Reference

Classes

- struct __gnu_pbds::container_error

  Base class for exceptions.
- struct __gnu_pbds::insert_error

  An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unabvailable, in which case the allocator_type's exception will be thrown).
- struct __gnu_pbds::join_error

  A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.
- struct __gnu_pbds::resize_error

  A container cannot be resized.

Namespaces

- namespace __gnu_pbds

Functions

- void **__gnu_pbds::__throw_container_error** ()
- void **__gnu_pbds::__throw_insert_error** ()
- void **__gnu_pbds::__throw_join_error** ()
- void **__gnu_pbds::__throw_resize_error** ()

5.196.1 Detailed Description

Contains exception classes.

Definition in file exception.hpp.

5.197 exception_defines.h File Reference

Macros

- #define **__catch**(X)
- #define **__throw_exception_again**
- #define **__try**

5.197.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

Definition in file exception_defines.h.

5.198 exception_ptr.h File Reference

Classes

- class std::__exception_ptr::exception_ptr

  An opaque pointer to an arbitrary exception.

Namespaces

- namespace std

Functions

- template<typename _Ex >
  exception_ptr std::copy_exception (_Ex __ex) noexcept
- exception_ptr std::current_exception () noexcept

- template<typename _Ex >
  exception_ptr std::make_exception_ptr (_Ex __ex) noexcept
- bool **std::__exception_ptr::operator!=** (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))
- bool **std::__exception_ptr::operator==** (const exception_ptr &, const exception_ptr &) noexcept __attribute__((__pure__))
- void std::rethrow_exception (exception_ptr) __attribute__((__noreturn__))
- void **std::__exception_ptr::swap** (exception_ptr &__lhs, exception_ptr &__rhs)

5.198.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

Definition in file exception_ptr.h.

5.199 extc++.h File Reference

5.199.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file extc++.h.

5.200 extptr_allocator.h File Reference

Classes

- class __gnu_cxx::_ExtPtr_allocator< _Tp >

  An example allocator which uses a non-standard pointer type.
  This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp >
  void **__gnu_cxx::swap** (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)

5.200.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

Definition in file extptr_allocator.h.

5.201 features.h File Reference

Macros

- #define _GLIBCXX_BAL_QUICKSORT
- #define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
- #define _GLIBCXX_FIND_EQUAL_SPLIT
- #define _GLIBCXX_FIND_GROWING_BLOCKS
- #define _GLIBCXX_MERGESORT
- #define _GLIBCXX_QUICKSORT
- #define _GLIBCXX_TREE_DYNAMIC_BALANCING
- #define _GLIBCXX_TREE_FULL_COPY
- #define _GLIBCXX_TREE_INITIAL_SPLITTING

5.201.1 Detailed Description

Defines on whether to include algorithm variants. Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file features.h.

5.201.2 Macro Definition Documentation

5.201.2.1 #define _GLIBCXX_BAL_QUICKSORT

Include parallel dynamically load-balanced quicksort.

See Also

__gnu_parallel::_Settings::sort_algorithm

Definition at line 55 of file features.h.

5.201.2.2 #define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS

Include the equal-sized blocks variant for std::find.

See Also

__gnu_parallel::_Settings::find_algorithm

Definition at line 67 of file features.h.

5.201.2.3 #define _GLIBCXX_FIND_EQUAL_SPLIT

Include the equal splitting variant for std::find.

See Also

__gnu_parallel::_Settings::find_algorithm

Definition at line 74 of file features.h.

5.201.2.4 #define _GLIBCXX_FIND_GROWING_BLOCKS

Include the growing blocks variant for std::find.

See Also

__gnu_parallel::_Settings::find_algorithm

Definition at line 61 of file features.h.

5.201.2.5 #define _GLIBCXX_MERGESORT

Include parallel multi-way mergesort.

See Also

__gnu_parallel::_Settings::sort_algorithm

Definition at line 41 of file features.h.

5.201.2.6 #define _GLIBCXX_QUICKSORT

Include parallel unbalanced quicksort.

See Also

__gnu_parallel::_Settings::sort_algorithm

Definition at line 48 of file features.h.

5.201.2.7 #define _GLIBCXX_TREE_DYNAMIC_BALANCING

Include the dynamic balancing variant for _Rb_tree::insert_unique(_IIter beg, _IIter __end).

See Also

__gnu_parallel::_Rb_tree

Definition at line 91 of file features.h.

5.201.2.8 #define _GLIBCXX_TREE_FULL_COPY

In order to sort the input sequence of _Rb_tree::insert_unique(_IIter beg, _IIter __end) a full copy of the input elements is done.

See Also

__gnu_parallel::_Rb_tree

Definition at line 100 of file features.h.

5.201.2.9 #define _GLIBCXX_TREE_INITIAL_SPLITTING

Include the initial splitting variant for _Rb_tree::insert_unique(_IIter beg, _IIter __end).

See Also

__gnu_parallel::_Rb_tree

Definition at line 83 of file features.h.

5.202 fenv.h File Reference

5.202.1 Detailed Description

This is a Standard C++ Library header.

Definition in file fenv.h.

5.203 find.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)
- template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)

5.203.1 Detailed Description

Parallel implementation base for std::find(), std::equal() and related functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file find.h.

5.204 find_fn_imps.hpp File Reference

5.204.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/find_fn_imps.hpp.

5.205 find_fn_imps.hpp File Reference

5.205.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_/find_fn_imps.hpp.

5.206 find_fn_imps.hpp File Reference

5.206.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/find_fn_imps.hpp.

5.207 find_fn_imps.hpp File Reference

5.207.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions.

Definition in file cc_hash_table_map_/find_fn_imps.hpp.

5.208 find_fn_imps.hpp File Reference

5.208.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions.

Definition in file gp_hash_table_map_/find_fn_imps.hpp.

5.209 find_fn_imps.hpp File Reference

5.209.1 Detailed Description

Contains implementations of lu_map_.

Definition in file list_update_map_/find_fn_imps.hpp.

5.210 find_fn_imps.hpp File Reference

5.210.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_/find_fn_imps.hpp.

5.211 find_fn_imps.hpp File Reference

5.211.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/find_fn_imps.hpp.

5.212 find_fn_imps.hpp File Reference

5.212.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/find_fn_imps.hpp.

5.213 find_fn_imps.hpp File Reference

5.213.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_tree_/find_fn_imps.hpp.

5.214 find_fn_imps.hpp File Reference

5.214.1 Detailed Description

Contains an implementation for thin_heap_.

Definition in file thin_heap_/find_fn_imps.hpp.

5.215 find_no_store_hash_fn_imps.hpp File Reference

5.215.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is not stored.

Definition in file find_no_store_hash_fn_imps.hpp.

5.216 find_selectors.h File Reference

Classes

- struct __gnu_parallel::__adjacent_find_selector

  Test predicate on two adjacent elements.
- struct __gnu_parallel::__find_first_of_selector< _FIterator >

  Test predicate on several elements.
- struct __gnu_parallel::__find_if_selector

  Test predicate on a single element, used for std::find() and std::find_if ().
- struct __gnu_parallel::__generic_find_selector

  Base class of all __gnu_parallel::__find_template selectors.

- struct __gnu_parallel::__mismatch_selector

  Test inverted predicate on a single element.

Namespaces

- namespace __gnu_parallel

5.216.1 Detailed Description

_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file find_selectors.h.

5.217 find_store_hash_fn_imps.hpp File Reference

5.217.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions, when the hash value is stored.

Definition in file cc_hash_table_map_/find_store_hash_fn_imps.hpp.

5.218 find_store_hash_fn_imps.hpp File Reference

5.218.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is stored.

Definition in file gp_hash_table_map_/find_store_hash_fn_imps.hpp.

5.219 for_each.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter , typename _UserOp , typename _Functionality , typename _Red , typename _Result >
  _UserOp __gnu_parallel::__for_each_template_random_access (_IIter __begin, _IIter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator_traits< _IIter >::difference_type __bound, _Parallelism __parallelism_tag)

5.219.1 Detailed Description

Main interface for embarrassingly parallel functions. The explicit implementation are in other header files, like workstealing.h, par_loop.h, omp_loop.h, and omp_loop_static.h. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file for_each.h.

5.220 for_each_selectors.h File Reference

Classes

- struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >

  General reduction, using a binary operator.
- struct __gnu_parallel::__accumulate_selector< _It >

  std::accumulate() selector.
- struct __gnu_parallel::__adjacent_difference_selector< _It >

  Selector that returns the difference between two adjacent __elements.
- struct __gnu_parallel::__count_if_selector< _It, _Diff >

  std::count_if () selector.
- struct __gnu_parallel::__count_selector< _It, _Diff >

  std::count() selector.
- struct __gnu_parallel::__fill_selector< _It >

  std::fill() selector.
- struct __gnu_parallel::__for_each_selector< _It >

  std::for_each() selector.
- struct __gnu_parallel::__generate_selector< _It >

  std::generate() selector.
- struct __gnu_parallel::__generic_for_each_selector< _It >

  Generic __selector for embarrassingly parallel functions.
- struct __gnu_parallel::__identity_selector< _It >

  Selector that just returns the passed iterator.
- struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >

  std::inner_product() selector.
- struct __gnu_parallel::__max_element_reduct< _Compare, _It >

  Reduction for finding the maximum element, using a comparator.
- struct __gnu_parallel::__min_element_reduct< _Compare, _It >

  Reduction for finding the maximum element, using a comparator.
- struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >

  std::replace() selector.
- struct __gnu_parallel::__replace_selector< _It, _Tp >

  std::replace() selector.
- struct __gnu_parallel::__transform1_selector< _It >

  std::transform() __selector, one input sequence variant.
- struct __gnu_parallel::__transform2_selector< _It >

  std::transform() __selector, two input sequences variant.
- struct __gnu_parallel::_DummyReduct

  Reduction function doing nothing.
- struct __gnu_parallel::_Nothing

  Functor doing nothing.

Namespaces

- namespace __gnu_parallel

5.220.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file for_each_selectors.h.

5.221 formatter.h File Reference

Namespaces

- namespace __gnu_debug

Enumerations

- enum **_Debug_msg_id** {
__msg_valid_range, **__msg_insert_singular**, **__msg_insert_different**, **__msg_erase_bad**,
__msg_erase_different, **__msg_subscript_oob**, **__msg_empty**, **__msg_unpartitioned**,
__msg_unpartitioned_pred, **__msg_unsorted**, **__msg_unsorted_pred**, **__msg_not_heap**,
__msg_not_heap_pred, **__msg_bad_bitset_write**, **__msg_bad_bitset_read**, **__msg_bad_bitset_flip**,
__msg_self_splice, **__msg_splice_alloc**, **__msg_splice_bad**, **__msg_splice_other**,
__msg_splice_overlap, **__msg_init_singular**, **__msg_init_copy_singular**, **__msg_init_const_singular**,
__msg_copy_singular, **__msg_bad_deref**, **__msg_bad_inc**, **__msg_bad_dec**,
__msg_iter_subscript_oob, **__msg_advance_oob**, **__msg_retreat_oob**, **__msg_iter_compare_bad**,
__msg_compare_different, **__msg_iter_order_bad**, **__msg_order_different**, **__msg_distance_bad**,
__msg_distance_different, **__msg_deref_istream**, **__msg_inc_istream**, **__msg_output_ostream**,
__msg_deref_istreambuf, **__msg_inc_istreambuf**, **__msg_insert_after_end**, **__msg_erase_after_bad**,
__msg_valid_range2, **__msg_local_iter_compare_bad**, **__msg_non_empty_range**, **__msg_self_move_assign**,
__msg_bucket_index_oob, **__msg_valid_load_factor**, **__msg_equal_allocs** }

Functions

- template<typename _Iterator >
bool **__gnu_debug::__check_singular** (_Iterator &)

5.221.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file formatter.h.

5.222 forward_list File Reference

Macros

- #define **_GLIBCXX_FORWARD_LIST**

5.222.1 Detailed Description

This is a Standard C++ Library header.

Definition in file forward_list.

5.223 forward_list File Reference

Classes

- class std::__debug::forward_list< _Tp, _Alloc >

  Class std::forward_list with safety/checking/debug instrumentation.

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_FORWARD_LIST**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator!=** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::__debug::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::__debug::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::__debug::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  void std::__debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)

5.223.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/forward_list.

5.224 forward_list File Reference

Classes

- class std::__profile::forward_list< _Tp, _Alloc >

  Class std::forward_list wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Macros

- #define **_GLIBCXX_PROFILE_FORWARD_LIST**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator!=** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator<** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::__profile::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::__profile::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::__profile::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  void std::__profile::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)

5.224.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file profile/forward_list.

5.225 forward_list.h File Reference

Classes

- struct std::_Fwd_list_base< _Tp, _Alloc >

  Base class for forward_list.

- struct std::_Fwd_list_const_iterator< _Tp >

  A forward_list::const_iterator.
- struct std::_Fwd_list_iterator< _Tp >

  A forward_list::iterator.
- struct std::_Fwd_list_node< _Tp >

  A helper node class for forward_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.
- struct std::_Fwd_list_node_base

  A helper basic node class for forward_list. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.
- class std::forward_list< _Tp, _Alloc >

  A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Namespaces

- namespace std

Functions

- template<typename _Tp >
  bool std::operator!= (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp >
  bool std::operator== (const _Fwd_list_iterator< _Tp > &__x, const _Fwd_list_const_iterator< _Tp > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp , typename _Alloc >
  void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly)

5.225.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <forward_list>.

Definition in file forward_list.h.

5.226 forward_list.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_FORWARD_LIST_TCC**

Functions

- template<typename _Tp , typename _Alloc >
  bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

5.226.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <forward_list>.

Definition in file forward_list.tcc.

5.227 fstream File Reference

Classes

- class std::basic_filebuf< _CharT, _Traits >

  The actual work of input and output (for files).
- class std::basic_fstream< _CharT, _Traits >

  Controlling input and output for files.
- class std::basic_ifstream< _CharT, _Traits >

  Controlling input for files.
- class std::basic_ofstream< _CharT, _Traits >

  Controlling output for files.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FSTREAM**

5.227.1 Detailed Description

This is a Standard C++ Library header.

Definition in file fstream.

5.228 fstream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_FSTREAM_TCC**

5.228.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <fstream>.

Definition in file fstream.tcc.

5.229 functexcept.h File Reference

Namespaces

- namespace std

Functions

- void **std::__throw_bad_alloc** (void) __attribute__((__noreturn__))
- void **std::__throw_bad_cast** (void) __attribute__((__noreturn__))
- void **std::__throw_bad_exception** (void) __attribute__((__noreturn__))
- void **std::__throw_bad_function_call** () __attribute__((__noreturn__))
- void **std::__throw_bad_typeid** (void) __attribute__((__noreturn__))
- void **std::__throw_domain_error** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_future_error** (int) __attribute__((__noreturn__))
- void **std::__throw_invalid_argument** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_ios_failure** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_length_error** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_logic_error** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_out_of_range** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_overflow_error** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_range_error** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_runtime_error** (const char ∗) __attribute__((__noreturn__))
- void **std::__throw_system_error** (int) __attribute__((__noreturn__))
- void **std::__throw_underflow_error** (const char ∗) __attribute__((__noreturn__))

5.229.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

This header provides support for -fno-exceptions.

Definition in file functexcept.h.

5.230 functional File Reference

Classes

- struct _Bind< _Signature >

  Type of the function object returned from bind().
- struct _Bind_result< _Result, _Signature >

  Type of the function object returned from bind<R>().
- class _Mu< _Arg, _IsBindExp, _IsPlaceholder >
- struct _Reference_wrapper_base_impl< _Unary, _Binary, _Tp >
- struct std::__is_location_invariant< _Tp >
- struct std::_Derives_from_binary_function< _Tp >

  Determines if the type _Tp derives from binary_function.
- struct std::_Derives_from_unary_function< _Tp >

  Determines if the type _Tp derives from unary_function.
- class std::_Function_base

  Base class of all polymorphic function object wrappers.
- struct std::_Maybe_get_result_type< _Has_result_type, _Functor >

  If we have found a result_type, extract it.
- struct std::_Maybe_unary_or_binary_function< _Res, _ArgTypes >
- struct std::_Maybe_unary_or_binary_function< _Res, _T1 >

  Derives from `unary_function`, as appropriate.
- struct std::_Maybe_unary_or_binary_function< _Res, _T1, _T2 >

  Derives from `binary_function`, as appropriate.
- struct std::_Maybe_wrap_member_pointer< _Tp >
- struct std::_Maybe_wrap_member_pointer< _Tp _Class::∗ >
- class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const >

  Implementation of `mem_fn` for const member function pointers.
- class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) const volatile >

  Implementation of `mem_fn` for const volatile member function pointers.
- class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...) volatile >

  Implementation of `mem_fn` for volatile member function pointers.
- class std::_Mem_fn< _Res(_Class::∗)(_ArgTypes...)>

  Implementation of `mem_fn` for member function pointers.
- class std::_Mu< _Arg, false, false >
- class std::_Mu< _Arg, false, true >
- class std::_Mu< _Arg, true, false >
- class std::_Mu< reference_wrapper< _Tp >, false, false >
- struct std::_Placeholder< _Num >

  The type of placeholder objects defined by libstdc++.
- struct std::_Reference_wrapper_base< _Tp >
- struct std::_Safe_tuple_element< __i, _Tuple >
- struct std::_Safe_tuple_element_impl< __i, _Tuple, _IsSafe >
- struct std::_Safe_tuple_element_impl< __i, _Tuple, false >
- struct std::_Weak_result_type< _Functor >
- struct std::_Weak_result_type_impl< _Functor >
- struct std::_Weak_result_type_impl< _Res(&)(_ArgTypes...)>

  Retrieve the result type for a function reference.
- struct std::_Weak_result_type_impl< _Res(∗)(_ArgTypes...)>

Retrieve the result type for a function pointer.

- struct std::_Weak_result_type_impl< _Res(_ArgTypes...)>

  Retrieve the result type for a function type.
- struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) const >

  Retrieve result type for a const member function pointer.
- struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) const volatile >

  Retrieve result type for a const volatile member function pointer.
- struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...) volatile >

  Retrieve result type for a volatile member function pointer.
- struct std::_Weak_result_type_impl< _Res(_Class::∗)(_ArgTypes...)>

  Retrieve result type for a member function pointer.
- class std::bad_function_call

  Exception class thrown when class template function's operator() is called with an empty target.
- class std::function< _Res(_ArgTypes...)>

  Primary class template for std::function.
  Polymorphic function wrapper.
- struct std::is_bind_expression< _Tp >

  Determines if the given type _Tp is a function object should be treated as a subexpression when evaluating calls to function objects returned by bind(). [TR1 3.6.1].
- struct std::is_bind_expression< _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_bind_expression< const _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_bind_expression< const volatile _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_bind_expression< volatile _Bind< _Signature > >

  Class template _Bind is always a bind expression.
- struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >

  Class template _Bind_result is always a bind expression.
- struct std::is_placeholder< _Tp >

  Determines if the given type _Tp is a placeholder in a bind() expression and, if so, which placeholder it is. [TR1 3.6.2].
- struct std::is_placeholder< _Placeholder< _Num > >
- class std::reference_wrapper< _Tp >

  Primary class template for reference_wrapper.

Namespaces

- namespace std
- namespace std::placeholders

Macros

- #define **_GLIBCXX_FUNCTIONAL**

Typedefs

- template<typename _Tp , typename _Tp2 = typename decay<_Tp>::type>
  using **std::__is_socketlike** = __or_< is_integral< _Tp2 >, is_enum< _Tp2 >>
- template<typename _Tp1 , typename _Tp2 >
  using **std::_NotSame** = __not_< is_same< typename std::decay< _Tp1 >::type, typename std::decay< _Tp2 >::type >>

Enumerations

- enum **_Manager_operation** { **__get_type_info**, **__get_functor_ptr**, **__clone_functor**, **__destroy_functor** }

Functions

- template<typename _Callable , typename... _Args>
  _Bind_simple_helper< _Callable, _Args...>::__type **std::__bind_simple** (_Callable &&__callable, _Args &&...__args)
- template<typename _Functor >
  _Functor & **std::__callable_functor** (_Functor &__f)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::∗ > **std::__callable_functor** (_Member _Class::∗&__p)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::∗ > **std::__callable_functor** (_Member _Class::∗const &__p)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::∗ > **std::__callable_functor** (_Member _Class::∗volatile &__p)
- template<typename _Member , typename _Class >
  _Mem_fn< _Member _Class::∗ > **std::__callable_functor** (_Member _Class::∗const volatile &__p)
- template<typename _Functor , typename... _Args>
  enable_if< (!is_member_pointer < _Functor >::value &&!is_function< _Functor > ::value &&!is_function < typename remove_pointer < _Functor >::type >::value), typename result_of< _Functor(_Args &&...)> ::type >::type std::__invoke (_Functor &__f, _Args &&...__args)
- template<typename _Functor , typename... _Args>
  enable_if< (is_pointer < _Functor >::value &&is_function< typename remove_pointer< _Functor > ::type >::value), typename result_of< _Functor(_Args &&...)> ::type >::type **std::__invoke** (_Functor __f, _Args &&...__args)
- template<std::size_t _Ind, typename... _Tp>
  auto **std::__volget** (volatile tuple< _Tp...> &__tuple) -> typename tuple_element< _Ind, tuple< _Tp...>>::type volatile &

- template<std::size_t _Ind, typename... _Tp>
  auto **std::__volget** (const volatile tuple< _Tp...> &__tuple) -> typename tuple_element< _Ind, tuple< _Tp...>>::type const volatile &
- template<typename _Func , typename... _BoundArgs>
  _Bind_helper< __is_socketlike
  < _Func >::value, _Func,
  _BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&...__args)
- template<typename _Result , typename _Func , typename... _BoundArgs>
  _Bindres_helper< _Result,
  _Func, _BoundArgs...>::type std::bind (_Func &&__f, _BoundArgs &&...__args)
- template<typename _Tp , typename _Class >
  _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept
- template<typename _Res , typename... _Args>
  bool std::operator!= (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _Res , typename... _Args>
  bool std::operator!= (nullptr_t, const function< _Res(_Args...)> &__f) noexcept
- template<typename _Res , typename... _Args>
  bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _Res , typename... _Args>
  bool std::operator== (nullptr_t, const function< _Res(_Args...)> &__f) noexcept
- template<typename _Res , typename... _Args>
  void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y)

- template<typename _Tp >
  reference_wrapper< _Tp > std::ref (_Tp &__t) noexcept
- template<typename _Tp >
  reference_wrapper< const _Tp > std::cref (const _Tp &__t) noexcept
- template<typename _Tp >
  void std::ref (const _Tp &&)=delete
- template<typename _Tp >
  void std::cref (const _Tp &&)=delete
- template<typename _Tp >
  reference_wrapper< _Tp > std::ref (reference_wrapper< _Tp > __t) noexcept
- template<typename _Tp >
  reference_wrapper< const _Tp > std::cref (reference_wrapper< _Tp > __t) noexcept

Variables

- const _Placeholder< 1 > **std::placeholders::_1**
- const _Placeholder< 10 > **std::placeholders::_10**
- const _Placeholder< 11 > **std::placeholders::_11**
- const _Placeholder< 12 > **std::placeholders::_12**
- const _Placeholder< 13 > **std::placeholders::_13**
- const _Placeholder< 14 > **std::placeholders::_14**
- const _Placeholder< 15 > **std::placeholders::_15**
- const _Placeholder< 16 > **std::placeholders::_16**
- const _Placeholder< 17 > **std::placeholders::_17**
- const _Placeholder< 18 > **std::placeholders::_18**
- const _Placeholder< 19 > **std::placeholders::_19**
- const _Placeholder< 2 > **std::placeholders::_2**
- const _Placeholder< 20 > **std::placeholders::_20**

- const _Placeholder< 21 > **std::placeholders::_21**
- const _Placeholder< 22 > **std::placeholders::_22**
- const _Placeholder< 23 > **std::placeholders::_23**
- const _Placeholder< 24 > **std::placeholders::_24**
- const _Placeholder< 25 > **std::placeholders::_25**
- const _Placeholder< 26 > **std::placeholders::_26**
- const _Placeholder< 27 > **std::placeholders::_27**
- const _Placeholder< 28 > **std::placeholders::_28**
- const _Placeholder< 29 > **std::placeholders::_29**
- const _Placeholder< 3 > **std::placeholders::_3**
- const _Placeholder< 4 > **std::placeholders::_4**
- const _Placeholder< 5 > **std::placeholders::_5**
- const _Placeholder< 6 > **std::placeholders::_6**
- const _Placeholder< 7 > **std::placeholders::_7**
- const _Placeholder< 8 > **std::placeholders::_8**
- const _Placeholder< 9 > **std::placeholders::_9**

5.230.1 Detailed Description

This is a Standard C++ Library header.

Definition in file functional.

5.231 functional File Reference

Classes

- class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >

  An SGI extension .
- struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >

  An SGI extension .
- struct __gnu_cxx::constant_unary_fun< _Result, _Argument >

  An SGI extension .
- struct __gnu_cxx::constant_void_fun< _Result >

  An SGI extension .
- struct __gnu_cxx::project1st< _Arg1, _Arg2 >

  An SGI extension .
- struct __gnu_cxx::project2nd< _Arg1, _Arg2 >

  An SGI extension .
- struct __gnu_cxx::select1st< _Pair >

  An SGI extension .
- struct __gnu_cxx::select2nd< _Pair >

  An SGI extension .
- class __gnu_cxx::subtractive_rng
- class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

  An SGI extension .

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_FUNCTIONAL**

Functions

- template< class _Operation1 , class _Operation2 >
  unary_compose< _Operation1,
  _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template< class _Operation1 , class _Operation2 , class _Operation3 >
  binary_compose< _Operation1,
  _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template< class _Result >
  constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)
- template< class _Result >
  constant_unary_fun< _Result,
  _Result > __gnu_cxx::constant1 (const _Result &__val)
- template< class _Result >
  constant_binary_fun< _Result,
  _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)
- template< class _Tp >
  _Tp __gnu_cxx::identity_element (std::plus< _Tp >)
- template< class _Tp >
  _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)
- template< class _Ret , class _Tp , class _Arg >
  mem_fun1_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1** (_Ret(_Tp::*__f)(_Arg))
- template< class _Ret , class _Tp , class _Arg >
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg))

5.231.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file ext/functional.

5.232 functional_hash.h File Reference

Classes

- struct hash< _Tp >

  Primary class template hash.
- struct std::hash< _Tp * >

  Partial specializations for pointer types.
- struct std::hash< bool >

  Explicit specialization for bool.

- struct std::hash< char >

  Explicit specialization for char.
- struct std::hash< char16_t >

  Explicit specialization for char16_t.
- struct std::hash< char32_t >

  Explicit specialization for char32_t.
- struct std::hash< double >

  Specialization for double.
- struct std::hash< float >

  Specialization for float.
- struct std::hash< int >

  Explicit specialization for int.
- struct std::hash< long >

  Explicit specialization for long.
- struct std::hash< long double >

  Specialization for long double.
- struct std::hash< long long >

  Explicit specialization for long long.
- struct std::hash< short >

  Explicit specialization for short.
- struct std::hash< signed char >

  Explicit specialization for signed char.
- struct std::hash< unsigned char >

  Explicit specialization for unsigned char.
- struct std::hash< unsigned int >

  Explicit specialization for unsigned int.
- struct std::hash< unsigned long >

  Explicit specialization for unsigned long.
- struct std::hash< unsigned long long >

  Explicit specialization for unsigned long long.
- struct std::hash< unsigned short >

  Explicit specialization for unsigned short.
- struct std::hash< wchar_t >

  Explicit specialization for wchar_t.

Namespaces

- namespace std

Macros

- #define **_Cxx_hashtable_define_trivial_hash**(_Tp)

5.232.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

Definition in file functional_hash.h.

5.233 functions.h File Reference

Namespaces

- namespace __gnu_debug

Functions

- template<typename _Iterator >
  _Siter_base< _Iterator >
  ::iterator_type __gnu_debug::__base (_Iterator __it)
- template<typename _Iterator >
  bool __gnu_debug::__check_dereferenceable (_Iterator &)
- template<typename _Tp >
  bool __gnu_debug::__check_dereferenceable (const _Tp ∗__ptr)
- template<typename _Iterator , typename _Sequence >
  bool __gnu_debug::__check_dereferenceable (const _Safe_iterator< _Iterator, _Sequence > &__x)
- template<typename _ForwardIterator , typename _Tp >
  bool **__gnu_debug::__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
  bool **__gnu_debug::__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator , typename _Tp >
  bool **__gnu_debug::__check_partitioned_lower_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp >
  bool **__gnu_debug::__check_partitioned_lower_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, std::random_access_iterator_tag __tag)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
  bool **__gnu_debug::__check_partitioned_lower_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp , typename _Pred >
  bool **__gnu_debug::__check_partitioned_lower_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, _Pred __pred, std::random_access_iterator_tag __tag)
- template<typename _ForwardIterator , typename _Tp >
  bool **__gnu_debug::__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
  bool **__gnu_debug::__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator , typename _Tp >
  bool **__gnu_debug::__check_partitioned_upper_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Tp >
  bool **__gnu_debug::__check_partitioned_upper_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, std::random_access_iterator_tag __tag)
- template<typename _ForwardIterator , typename _Tp , typename _Pred >
  bool **__gnu_debug::__check_partitioned_upper_aux** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred, std::forward_iterator_tag)

- template<typename _Iterator , typename _Sequence , typename _Tp , typename _Pred >
  bool **__gnu_debug::__check_partitioned_upper_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, const _Tp &__value, _Pred __pred, std::random_access_iterator_tag __tag)
- template<typename _Iterator >
  bool **__gnu_debug::__check_singular** (_Iterator &)
- template<typename _Tp >
  bool __gnu_debug::__check_singular (const _Tp *__ptr)
- template<typename _Iterator , typename _Sequence >
  bool __gnu_debug::__check_singular (const _Safe_iterator< _Iterator, _Sequence > &__x)
- bool **__gnu_debug::__check_singular_aux** (const void *)
- template<typename _InputIterator >
  bool **__gnu_debug::__check_sorted** (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _InputIterator , typename _Predicate >
  bool **__gnu_debug::__check_sorted** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)
- template<typename _InputIterator >
  bool **__gnu_debug::__check_sorted_aux** (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _ForwardIterator >
  bool **__gnu_debug::__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence >
  bool **__gnu_debug::__check_sorted_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, std::random_access_iterator_tag __tag)
- template<typename _InputIterator , typename _Predicate >
  bool **__gnu_debug::__check_sorted_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)
- template<typename _ForwardIterator , typename _Predicate >
  bool **__gnu_debug::__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)
- template<typename _Iterator , typename _Sequence , typename _Predicate >
  bool **__gnu_debug::__check_sorted_aux** (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last, _Predicate __pred, std::random_access_iterator_tag __tag)
- template<typename _InputIterator1 , typename _InputIterator2 >
  bool **__gnu_debug::__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _Predicate >
  bool **__gnu_debug::__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)
- template<typename _InputIterator >
  bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)
- template<typename _InputIterator >
  bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _InputIterator , typename _Predicate >
  bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)
- template<typename _InputIterator , typename _Predicate >
  bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)
- template<typename _CharT , typename _Integer >
  const _CharT * __gnu_debug::__check_string (const _CharT *__s, const _Integer &__n __attribute__((__unused__)))

- template<typename _CharT >
  const _CharT ∗ __gnu_debug::__check_string (const _CharT ∗__s)
- template<typename _InputIterator >
  _InputIterator **__gnu_debug::__check_valid_range** (const _InputIterator &__first, const _InputIterator &__last __attribute__((__unused__)))
- template<typename _InputIterator >
  bool __gnu_debug::__valid_range (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _Iterator , typename _Sequence >
  bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence > &__first, const _Safe_iterator< _Iterator, _Sequence > &__last)
- template<typename _Iterator , typename _Sequence >
  bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _Sequence > &__first, const _Safe_local_iterator< _Iterator, _Sequence > &__last)
- template<typename _Integral >
  bool __gnu_debug::__valid_range_aux (const _Integral &, const _Integral &, std::__true_type)
- template<typename _InputIterator >
  bool __gnu_debug::__valid_range_aux (const _InputIterator &__first, const _InputIterator &__last, std::__false_type)
- template<typename _RandomAccessIterator >
  bool __gnu_debug::__valid_range_aux2 (const _RandomAccessIterator &__first, const _RandomAccessIterator &__last, std::random_access_iterator_tag)
- template<typename _InputIterator >
  bool __gnu_debug::__valid_range_aux2 (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)

5.233.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file functions.h.

5.234 future File Reference

Classes

- class std::future_error

  Exception type thrown by futures.
- struct std::is_error_code_enum< future_errc >

  Specialization.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FUTURE**

Enumerations

- enum std::future_errc
- enum std::future_status
- enum std::launch

Functions

- template<typename _Fn , typename... _Args>
  future< typename result_of
  < _Fn(_Args...)>::type > **std::async** (launch __policy, _Fn &&__fn, _Args &&...__args)
- template<typename _Fn , typename... _Args>
  future< typename result_of
  < _Fn(_Args...)>::type > **std::async** (_Fn &&__fn, _Args &&...__args)
- const error_category & std::future_category () noexcept
- error_code std::make_error_code (future_errc __errc) noexcept
- error_condition std::make_error_condition (future_errc __errc) noexcept
- constexpr launch **std::operator&** (launch __x, launch __y)
- launch & **std::operator&=** (launch &__x, launch __y)
- constexpr launch **std::operator^** (launch __x, launch __y)
- launch & **std::operator^=** (launch &__x, launch __y)
- constexpr launch **std::operator|** (launch __x, launch __y)
- launch & **std::operator|=** (launch &__x, launch __y)
- constexpr launch **std::operator~** (launch __x)

5.234.1 Detailed Description

This is a Standard C++ Library header.

Definition in file future.

5.235 gp_ht_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_GP_HASH_NAME**
- #define **PB_DS_GP_HASH_TRAITS_BASE**
- #define **PB_DS_HASH_EQ_FN_C_DEC**
- #define **PB_DS_RANGED_PROBE_FN_C_DEC**

Variables

- **empty_entry_status**
- **erased_entry_status**
- **valid_entry_status**

5.235.1 Detailed Description

Contains an implementation class for general probing hash.

Definition in file gp_ht_map_.hpp.

5.236 gslice.h File Reference

Classes

- class std::gslice

  Class defining multi-dimensional subset of an array.

Namespaces

- namespace std

5.236.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file gslice.h.

5.237 gslice_array.h File Reference

Classes

- class std::gslice_array< _Tp >

  Reference to multi-dimensional subset of an array.

Namespaces

- namespace std

Macros

- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)

5.237.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file gslice_array.h.

5.238 hash_bytes.h File Reference

Namespaces

- namespace std

Functions

- size_t **std::_Fnv_hash_bytes** (const void ∗__ptr, size_t __len, size_t __seed)
- size_t **std::_Hash_bytes** (const void ∗__ptr, size_t __len, size_t __seed)

5.238.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

Definition in file hash_bytes.h.

5.239 hash_eq_fn.hpp File Reference

Classes

- struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >

  Specialization 1 - The client requests that hash values not be stored.
- struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >

  Specialization 2 - The client requests that hash values be stored.
- struct hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >

  Primary template.

Namespaces

- namespace __gnu_pbds

5.239.1 Detailed Description

Contains 2 eqivalence functions, one employing a hash value, and one ignoring it.

Definition in file hash_eq_fn.hpp.

5.240 hash_exponential_size_policy_imp.hpp File Reference

5.240.1 Detailed Description

Contains a resize size policy implementation.

Definition in file hash_exponential_size_policy_imp.hpp.

5.241 hash_fun.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- size_t **__gnu_cxx::__stl_hash_string** (const char ∗__s)

5.241.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file hash_fun.h.

5.242 hash_load_check_resize_trigger_imp.hpp File Reference

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_ASSERT_VALID**(X)

5.242.1 Detailed Description

Contains a resize trigger implementation.

Definition in file hash_load_check_resize_trigger_imp.hpp.

5.243 hash_load_check_resize_trigger_size_base.hpp File Reference

Classes

- class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >

  Specializations.
- class hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >

  Primary template.

Namespaces

- namespace __gnu_pbds

5.243.1 Detailed Description

Contains an base holding size for some resize policies.

Definition in file hash_load_check_resize_trigger_size_base.hpp.

5.244 hash_map File Reference

Classes

- class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_BACKWARD_HASH_MAP**

Functions

- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  bool **__gnu_cxx::operator!=** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key , class _Tp , class _HF , class _EqKey , class _Alloc >
  bool **__gnu_cxx::operator!=** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  bool **__gnu_cxx::operator==** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key , class _Tp , class _HF , class _EqKey , class _Alloc >
  bool **__gnu_cxx::operator==** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  void **__gnu_cxx::swap** (hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key , class _Tp , class _HashFn , class _EqlKey , class _Alloc >
  void **__gnu_cxx::swap** (hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)

5.244.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file hash_map.

5.245 hash_policy.hpp File Reference

Classes

- class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >

    A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.
- class __gnu_pbds::direct_mask_range_hashing< Size_Type >

    A mask range-hashing class (uses a bitmask).
- class __gnu_pbds::direct_mod_range_hashing< Size_Type >

    A mod range-hashing class (uses the modulo function).
- class __gnu_pbds::hash_exponential_size_policy< Size_Type >

    A size policy whose sequence of sizes form an exponential sequence (typically powers of 2.
- class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >

    A resize trigger policy based on a load check. It keeps the load factor between some load factors load_min and load_max.
- class __gnu_pbds::hash_prime_size_policy

    A size policy whose sequence of sizes form a nearly-exponential sequence of primes.
- class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >

    A resize policy which delegates operations to size and trigger policies.
- class __gnu_pbds::linear_probe_fn< Size_Type >

    A probe sequence policy using fixed increments.
- class __gnu_pbds::quadratic_probe_fn< Size_Type >

    A probe sequence policy using square increments.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_SIZE_BASE_C_DEC**

Enumerations

- enum { **num_distinct_sizes_32_bit**, **num_distinct_sizes_64_bit**, **num_distinct_sizes** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_sizes** [num_distinct_sizes_64_bit]

5.245.1 Detailed Description

Contains hash-related policies.

Definition in file hash_policy.hpp.

5.246 hash_prime_size_policy_imp.hpp File Reference

Enumerations

- enum { **num_distinct_sizes_32_bit**, **num_distinct_sizes_64_bit**, **num_distinct_sizes** }

Variables

- static const std::size_t **detail::g_a_sizes** [num_distinct_sizes_64_bit]

5.246.1 Detailed Description

Contains a resize size policy implementation.

Definition in file hash_prime_size_policy_imp.hpp.

5.247 hash_set File Reference

Classes

- class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_BACKWARD_HASH_SET**

Functions

- template<class _Value , class _HashFcn , class _EqualKey , class _Alloc >
  bool **__gnu_cxx::operator!=** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  bool **__gnu_cxx::operator!=** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value , class _HashFcn , class _EqualKey , class _Alloc >
  bool **__gnu_cxx::operator==** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  bool **__gnu_cxx::operator==** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  void **__gnu_cxx::swap** (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >
  void **__gnu_cxx::swap** (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)

5.247.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file hash_set.

5.248 hash_standard_resize_policy_imp.hpp File Reference

5.248.1 Detailed Description

Contains a resize policy implementation.

Definition in file hash_standard_resize_policy_imp.hpp.

5.249 hashtable.h File Reference

Classes

- class std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Namespaces

- namespace std

Typedefs

- template<typename _Tp , typename _Hash >
  using **std::__cache_default** = __not_< __and_< __is_fast_hash< _Hash >, is_default_constructible< _Hash >, is_copy_assignable< _Hash >, __detail::__is_noexcept_hash< _Tp, _Hash >>>

5.249.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map> or < unordered_set>.

Definition in file bits/hashtable.h.

5.250 hashtable.h File Reference

Namespaces

- namespace __gnu_cxx

Enumerations

- enum { **_S_num_primes** }

Functions

- unsigned long **__gnu_cxx::__stl_next_prime** (unsigned long __n)
- template<class _Val , class _Key , class _HF , class _Ex , class _Eq , class _All >
  bool **__gnu_cxx::operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val , class _Key , class _HF , class _Ex , class _Eq , class _All >
  bool **__gnu_cxx::operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val , class _Key , class _HF , class _Extract , class _EqKey , class _All >
  void **__gnu_cxx::swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)

5.250.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file backward/hashtable.h.

5.251 hashtable_policy.h File Reference

Classes

- struct _Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, __cache_hash_code >
- struct _Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
- struct _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct _Hash_node< _Value, _Cache_hash_code >
- struct _Hashtable_ebo_helper< _Nm, _Tp, __use_ebo >
- struct _Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators, _Unique_keys >
- struct _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
- struct _Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

- struct std::__detail::_Before_begin< _NodeAlloc >
- struct std::__detail::_Default_ranged_hash

  Default ranged hash function H. In principle it should be a function object composed from objects of type H1 and H2 such that h(k, N) = h2(h1(k), N), but that would mean making extra copies of h1 and h2. So instead we'll just use a tag to tell class template hashtable to do that composition.
- struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, false >

  Specialization.
- struct std::__detail::_Equal_helper< _Key, _Value, _ExtractKey, _Equal, _HashCodeType, true >

  Specialization.
- struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

  Specialization.
- struct std::__detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

  Specialization.
- struct std::__detail::_Equality_base
- struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >

  Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.
- struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >

  Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.
- struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

  Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.
- struct std::__detail::_Hash_node< _Value, false >
- struct std::__detail::_Hash_node< _Value, true >
- struct std::__detail::_Hash_node_base
- struct std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >
- struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, false >

  Specialization not using EBO.
- struct std::__detail::_Hashtable_ebo_helper< _Nm, _Tp, true >

  Specialization using EBO.
- struct std::__detail::_Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
- struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false, _Unique_keys >

  Specialization.
- struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, false >

  Specialization.
- struct std::__detail::_Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true, true >

  Specialization.
- struct std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
- struct std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

  local const_iterators
- struct std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >

local iterators

- struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >

    Specialization.
- struct std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >

    Specialization.
- struct std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
- struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >

    Partial specialization, __unique_keys set to false.
- struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >

    Partial specialization, __unique_keys set to true.
- struct std::__detail::_Mod_range_hashing

    Default range hashing function: use division to fold a large number into the range [0, N).
- struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >

    Node const_iterators, used to iterate through all the hashtable.
- struct std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >

    Node iterators, used to iterate through all the hashtable.
- struct std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >

    Base class for node iterators.
- struct std::__detail::_Prime_rehash_policy

    Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.
- struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _Prime_rehash_policy, _Traits >

    Specialization.
- class std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **std::__detail::__distance_fw** (_Iterator __first, _Iterator __last, std::input_iterator_tag)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **std::__detail::__distance_fw** (_Iterator __first, _Iterator __last, std::forward_iterator_tag)
- template<class _Iterator >
  std::iterator_traits
  < _Iterator >::difference_type **std::__detail::__distance_fw** (_Iterator __first, _Iterator __last)
- template<typename _Value , bool _Cache_hash_code>
  bool **std::__detail::operator!=** (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)

- template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>
  bool **std::__detail::operator!=** (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)
- template<typename _Value , bool _Cache_hash_code>
  bool **std::__detail::operator==** (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y)
- template<typename _Key , typename _Value , typename _ExtractKey , typename _H1 , typename _H2 , typename _Hash , bool __cache>
  bool **std::__detail::operator==** (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)

5.251.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map> or <unordered_set>.

Definition in file hashtable_policy.h.

5.252 indirect_array.h File Reference

Classes

- class std::indirect_array< _Tp >

  Reference to arbitrary subset of an array.

Namespaces

- namespace std

Macros

- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)

5.252.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file indirect_array.h.

5.253 info_fn_imps.hpp File Reference

5.253.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/info_fn_imps.hpp.

5.254 info_fn_imps.hpp File Reference

5.254.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/info_fn_imps.hpp.

5.255 info_fn_imps.hpp File Reference

5.255.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container info related functions.

Definition in file cc_hash_table_map_/info_fn_imps.hpp.

5.256 info_fn_imps.hpp File Reference

5.256.1 Detailed Description

Contains implementations of gp_ht_map_'s entire container info related functions.

Definition in file gp_hash_table_map_/info_fn_imps.hpp.

5.257 info_fn_imps.hpp File Reference

5.257.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/info_fn_imps.hpp.

5.258 info_fn_imps.hpp File Reference

5.258.1 Detailed Description

Contains implementations of lu_map_.

Definition in file list_update_map_/info_fn_imps.hpp.

5.259 info_fn_imps.hpp File Reference

5.259.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/info_fn_imps.hpp.

5.260 info_fn_imps.hpp File Reference

5.260.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/info_fn_imps.hpp.

5.261 info_fn_imps.hpp File Reference

5.261.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/info_fn_imps.hpp.

5.262 info_fn_imps.hpp File Reference

5.262.1 Detailed Description

Contains an implementation.

Definition in file splay_tree_/info_fn_imps.hpp.

5.263 initializer_list File Reference

Classes

- class std::initializer_list< _E >

  initializer_list

Namespaces

- namespace std

Functions

- template<class _Tp >
  constexpr const _Tp ∗ std::begin (initializer_list< _Tp > __ils) noexcept
- template<class _Tp >
  constexpr const _Tp ∗ std::end (initializer_list< _Tp > __ils) noexcept

5.263.1 Detailed Description

This is a Standard C++ Library header.

Definition in file initializer_list.

5.264 insert_fn_imps.hpp File Reference

5.264.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/insert_fn_imps.hpp.

5.265 insert_fn_imps.hpp File Reference

5.265.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_/insert_fn_imps.hpp.

5.266 insert_fn_imps.hpp File Reference

5.266.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/insert_fn_imps.hpp.

5.267 insert_fn_imps.hpp File Reference

5.267.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions.

Definition in file cc_hash_table_map_/insert_fn_imps.hpp.

5.268 insert_fn_imps.hpp File Reference

5.268.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions.

Definition in file gp_hash_table_map_/insert_fn_imps.hpp.

5.269 insert_fn_imps.hpp File Reference

5.269.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/insert_fn_imps.hpp.

5.270 insert_fn_imps.hpp File Reference

5.270.1 Detailed Description

Contains implementations of lu_map_.

Definition in file list_update_map_/insert_fn_imps.hpp.

5.271 insert_fn_imps.hpp File Reference

5.271.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/insert_fn_imps.hpp.

5.272 insert_fn_imps.hpp File Reference

5.272.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_/insert_fn_imps.hpp.

5.273 insert_fn_imps.hpp File Reference

5.273.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/insert_fn_imps.hpp.

5.274 insert_fn_imps.hpp File Reference

5.274.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

Definition in file rc_binomial_heap_/insert_fn_imps.hpp.

5.275 insert_fn_imps.hpp File Reference

5.275.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_tree_/insert_fn_imps.hpp.

5.276 insert_fn_imps.hpp File Reference

5.276.1 Detailed Description

Contains an implementation for thin_heap_.

Definition in file thin_heap_/insert_fn_imps.hpp.

5.277 insert_join_fn_imps.hpp File Reference

5.277.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file insert_join_fn_imps.hpp.

5.278 insert_no_store_hash_fn_imps.hpp File Reference

5.278.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions, when the hash value is not stored.

Definition in file cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp.

5.279 insert_no_store_hash_fn_imps.hpp File Reference

5.279.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is not stored.

Definition in file gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp.

5.280 insert_store_hash_fn_imps.hpp File Reference

5.280.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions, when the hash value is stored.

Definition in file cc_hash_table_map_/insert_store_hash_fn_imps.hpp.

5.281 insert_store_hash_fn_imps.hpp File Reference

5.281.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is stored.

Definition in file gp_hash_table_map_/insert_store_hash_fn_imps.hpp.

5.282 iomanip File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_IOMANIP**

Functions

- template<typename _MoneyT >
  _Get_money< _MoneyT > std::get_money (_MoneyT &__mon, bool __intl=false)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)
- template<typename _CharT , typename _Traits , typename _MoneyT >
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)
- template<typename _CharT , typename _Traits , typename _MoneyT >
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)
- template<typename _MoneyT >
  _Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __intl=false)
- _Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)
- _Setbase std::setbase (int __base)
- template<typename _CharT >
  _Setfill< _CharT > std::setfill (_CharT __c)
- _Setiosflags std::setiosflags (ios_base::fmtflags __mask)
- _Setprecision std::setprecision (int __n)
- _Setw std::setw (int __n)

5.282.1 Detailed Description

This is a Standard C++ Library header.

Definition in file iomanip.

5.283 ios File Reference

Macros

- #define **_GLIBCXX_IOS**

5.283.1 Detailed Description

This is a Standard C++ Library header.

Definition in file ios.

5.284 ios_base.h File Reference

Classes

- class std::ios_base

  The base of the I/O class hierarchy.
  This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` *when they need to specify the full name of the various I/O flags (e.g., the openmodes).*
- class std::ios_base::failure

  These are thrown to indicate problems with io.
  27.4.2.1.1 Class ios_base::failure.

Namespaces

- namespace std

Enumerations

- enum **_Ios_Fmtflags** {
  _S_boolalpha, **_S_dec**, **_S_fixed**, **_S_hex**,
  _S_internal, **_S_left**, **_S_oct**, **_S_right**,
  _S_scientific, **_S_showbase**, **_S_showpoint**, **_S_showpos**,
  _S_skipws, **_S_unitbuf**, **_S_uppercase**, **_S_adjustfield**,
  _S_basefield, **_S_floatfield**, **_S_ios_fmtflags_end** }
- enum **_Ios_Iostate** {
  _S_goodbit, **_S_badbit**, **_S_eofbit**, **_S_failbit**,
  _S_ios_iostate_end }
- enum **_Ios_Openmode** {
  _S_app, **_S_ate**, **_S_bin**, **_S_in**,
  _S_out, **_S_trunc**, **_S_ios_openmode_end** }
- enum **_Ios_Seekdir** { **_S_beg**, **_S_cur**, **_S_end**, **_S_ios_seekdir_end** }

Functions

- ios_base & std::boolalpha (ios_base &__base)
- ios_base & std::dec (ios_base &__base)
- ios_base & std::fixed (ios_base &__base)
- ios_base & std::hex (ios_base &__base)
- ios_base & std::internal (ios_base &__base)
- ios_base & std::left (ios_base &__base)
- ios_base & std::noboolalpha (ios_base &__base)
- ios_base & std::noshowbase (ios_base &__base)
- ios_base & std::noshowpoint (ios_base &__base)
- ios_base & std::noshowpos (ios_base &__base)
- ios_base & std::noskipws (ios_base &__base)
- ios_base & std::nounitbuf (ios_base &__base)
- ios_base & std::nouppercase (ios_base &__base)
- ios_base & std::oct (ios_base &__base)
- constexpr _Ios_Fmtflags **std::operator&** (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- constexpr _Ios_Openmode **std::operator&** (_Ios_Openmode __a, _Ios_Openmode __b)
- constexpr _Ios_Iostate **std::operator&** (_Ios_Iostate __a, _Ios_Iostate __b)
- const _Ios_Fmtflags & **std::operator&=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- const _Ios_Openmode & **std::operator&=** (_Ios_Openmode &__a, _Ios_Openmode __b)
- const _Ios_Iostate & **std::operator&=** (_Ios_Iostate &__a, _Ios_Iostate __b)
- constexpr _Ios_Fmtflags **std::operator^** (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- constexpr _Ios_Openmode **std::operator^** (_Ios_Openmode __a, _Ios_Openmode __b)
- constexpr _Ios_Iostate **std::operator^** (_Ios_Iostate __a, _Ios_Iostate __b)
- const _Ios_Fmtflags & **std::operator^=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- const _Ios_Openmode & **std::operator^=** (_Ios_Openmode &__a, _Ios_Openmode __b)
- const _Ios_Iostate & **std::operator^=** (_Ios_Iostate &__a, _Ios_Iostate __b)
- constexpr _Ios_Fmtflags **std::operator|** (_Ios_Fmtflags __a, _Ios_Fmtflags __b)
- constexpr _Ios_Openmode **std::operator|** (_Ios_Openmode __a, _Ios_Openmode __b)
- constexpr _Ios_Iostate **std::operator|** (_Ios_Iostate __a, _Ios_Iostate __b)
- const _Ios_Fmtflags & **std::operator|=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b)
- const _Ios_Openmode & **std::operator|=** (_Ios_Openmode &__a, _Ios_Openmode __b)
- const _Ios_Iostate & **std::operator|=** (_Ios_Iostate &__a, _Ios_Iostate __b)
- constexpr _Ios_Fmtflags **std::operator~** (_Ios_Fmtflags __a)
- constexpr _Ios_Openmode **std::operator~** (_Ios_Openmode __a)
- constexpr _Ios_Iostate **std::operator~** (_Ios_Iostate __a)
- ios_base & std::right (ios_base &__base)
- ios_base & std::scientific (ios_base &__base)
- ios_base & std::showbase (ios_base &__base)
- ios_base & std::showpoint (ios_base &__base)
- ios_base & std::showpos (ios_base &__base)
- ios_base & std::skipws (ios_base &__base)
- ios_base & std::unitbuf (ios_base &__base)
- ios_base & std::uppercase (ios_base &__base)

5.284.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

Definition in file ios_base.h.

5.285 iosfwd File Reference

Classes

- class std::basic_filebuf< _CharT, _Traits >

  The actual work of input and output (for files).
- class std::basic_fstream< _CharT, _Traits >

  Controlling input and output for files.
- class std::basic_ifstream< _CharT, _Traits >

  Controlling input for files.
- class std::basic_ofstream< _CharT, _Traits >

  Controlling output for files.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_IOSFWD**

Typedefs

- typedef basic_filebuf< char > std::filebuf
- typedef basic_fstream< char > std::fstream
- typedef basic_ifstream< char > std::ifstream
- typedef basic_ios< char > std::ios
- typedef basic_iostream< char > std::iostream
- typedef basic_istream< char > std::istream
- typedef basic_istringstream< char > std::istringstream
- typedef basic_ofstream< char > std::ofstream
- typedef basic_ostream< char > std::ostream
- typedef basic_ostringstream< char > std::ostringstream
- typedef basic_streambuf< char > std::streambuf
- typedef basic_stringbuf< char > std::stringbuf
- typedef basic_stringstream< char > std::stringstream
- typedef basic_filebuf< wchar_t > std::wfilebuf
- typedef basic_fstream< wchar_t > std::wfstream
- typedef basic_ifstream< wchar_t > std::wifstream
- typedef basic_ios< wchar_t > std::wios
- typedef basic_iostream< wchar_t > std::wiostream
- typedef basic_istream< wchar_t > std::wistream
- typedef basic_istringstream< wchar_t > std::wistringstream
- typedef basic_ofstream< wchar_t > std::wofstream
- typedef basic_ostream< wchar_t > std::wostream
- typedef basic_ostringstream< wchar_t > std::wostringstream
- typedef basic_streambuf< wchar_t > std::wstreambuf
- typedef basic_stringbuf< wchar_t > std::wstringbuf
- typedef basic_stringstream< wchar_t > std::wstringstream

5.285.1 Detailed Description

This is a Standard C++ Library header.

Definition in file iosfwd.

5.286 iostream File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_IOSTREAM**

Variables

- static ios_base::Init std::__ioinit

Standard Stream Objects

The <iostream> header declares the eight standard stream objects. For other declarations, see `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt11ch24.html` *and the I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` *streams, and to be available during program startup and termination. For more information, see the HOWTO linked to above.*

- istream std::cin
- ostream std::cout
- ostream std::cerr
- ostream std::clog
- wistream std::wcin
- wostream std::wcout
- wostream std::wcerr
- wostream std::wclog

5.286.1 Detailed Description

This is a Standard C++ Library header.

Definition in file iostream.

5.287 istream File Reference

Classes

- class std::basic_iostream< _CharT, _Traits >

  Template class basic_iostream.
- class std::basic_istream< _CharT, _Traits >

  Template class basic_istream.
- class std::basic_istream< _CharT, _Traits >::sentry

  Performs setup work for input streams.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ISTREAM**

Functions

- template<typename _CharT , typename _Traits , typename _Tp >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &&__is, _Tp &__x)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)

- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<class _Traits >
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)
- template<class _Traits >
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)

- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT ∗__s)
- template<>
  basic_istream< char > & std::operator>> (basic_istream< char > &__in, char ∗__s)
- template<class _Traits >
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char ∗__s)
- template<class _Traits >
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char ∗__s)

5.287.1 Detailed Description

This is a Standard C++ Library header.

Definition in file istream.

5.288 istream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_ISTREAM_TCC**

Functions

- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<typename _CharT , typename _Traits >
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)

5.288.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <istream>.

Definition in file istream.tcc.

5.289 iterator File Reference

Macros

- #define **_GLIBCXX_ITERATOR**

5.289.1 Detailed Description

This is a Standard C++ Library header.

Definition in file iterator.

5.290 iterator File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_ITERATOR**

Functions

- template<typename _InputIterator , typename _Distance >
  void **__gnu_cxx::__distance** (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Distance >
  void **__gnu_cxx::__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag)
- template<typename _InputIterator , typename _Distance >
  void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)

5.290.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file ext/iterator.

5.291 iterator.h File Reference

Classes

- class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >

  A pair of iterators. The usual iterator operations are applied to both child iterators.
- class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >

  A triple of iterators. The usual iterator operations are applied to all three child iterators.

Namespaces

- namespace __gnu_parallel

5.291.1 Detailed Description

Helper iterator classes for the std::transform() functions. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file iterator.h.

5.292 iterator.hpp File Reference

Classes

- class iterator_

  Range-type iterator.

5.292.1 Detailed Description

Contains an iterator_ class used for ranging over the elements of the table.

Definition in file iterator.hpp.

5.293 iterator_fn_imps.hpp File Reference

5.293.1 Detailed Description

Contains implementations of gp_ht_map_'s iterators related functions, e.g., begin().

Definition in file iterator_fn_imps.hpp.

5.294 iterator_tracker.h File Reference

Namespaces

- namespace std
- namespace std::__profile

Functions

- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **std::__profile::operator!=** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **std::__profile::operator!=** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  __iterator_tracker< _Iterator,
  _Sequence > **std::__profile::operator+** (typename __iterator_tracker< _Iterator, _Sequence >::difference_type __n, const __iterator_tracker< _Iterator, _Sequence > &__i)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  __iterator_tracker< _IteratorL,
  _Sequence >::difference_type **std::__profile::operator-** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  __iterator_tracker< _Iterator,
  _Sequence >::difference_type **std::__profile::operator-** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **std::__profile::operator<** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **std::__profile::operator<** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **std::__profile::operator<=** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **std::__profile::operator<=** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **std::__profile::operator==** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **std::__profile::operator==** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **std::__profile::operator>** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **std::__profile::operator>** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)

- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **std::__profile::operator>=** (const __iterator_tracker< _IteratorL, _Sequence > &__lhs, const __iterator_tracker< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **std::__profile::operator>=** (const __iterator_tracker< _Iterator, _Sequence > &__lhs, const __iterator_tracker< _Iterator, _Sequence > &__rhs)

5.294.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file iterator_tracker.h.

5.295 iterators_fn_imps.hpp File Reference

5.295.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/iterators_fn_imps.hpp.

5.296 iterators_fn_imps.hpp File Reference

5.296.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/iterators_fn_imps.hpp.

5.297 iterators_fn_imps.hpp File Reference

5.297.1 Detailed Description

Contains implementations of cc_ht_map_'s iterators related functions, e.g., begin().

Definition in file cc_hash_table_map_/iterators_fn_imps.hpp.

5.298 iterators_fn_imps.hpp File Reference

5.298.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/iterators_fn_imps.hpp.

5.299 iterators_fn_imps.hpp File Reference

5.299.1 Detailed Description

Contains implementations of lu_map_.

Definition in file list_update_map_/iterators_fn_imps.hpp.

5.300 iterators_fn_imps.hpp File Reference

5.300.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/iterators_fn_imps.hpp.

5.301 iterators_fn_imps.hpp File Reference

5.301.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/iterators_fn_imps.hpp.

5.302 left_child_next_sibling_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

  Base class for a basic heap.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.302.1 Detailed Description

Contains an implementation class for a basic heap.

Definition in file left_child_next_sibling_heap_.hpp.

5.303 limits File Reference

Classes

- struct std::__numeric_limits_base

  Part of std::numeric_limits.
- struct std::numeric_limits< _Tp >

  Properties of fundamental types.
- struct std::numeric_limits< bool >

  numeric_limits<bool> specialization.
- struct std::numeric_limits< char >

numeric_limits<char> specialization.

- struct std::numeric_limits< char16_t >

  numeric_limits<char16_t> specialization.
- struct std::numeric_limits< char32_t >

  numeric_limits<char32_t> specialization.
- struct std::numeric_limits< double >

  numeric_limits<double> specialization.
- struct std::numeric_limits< float >

  numeric_limits<float> specialization.
- struct std::numeric_limits< int >

  numeric_limits<int> specialization.
- struct std::numeric_limits< long >

  numeric_limits<long> specialization.
- struct std::numeric_limits< long double >

  numeric_limits<long double> specialization.
- struct std::numeric_limits< long long >

  numeric_limits<long long> specialization.
- struct std::numeric_limits< short >

  numeric_limits<short> specialization.
- struct std::numeric_limits< signed char >

  numeric_limits<signed char> specialization.
- struct std::numeric_limits< unsigned char >

  numeric_limits<unsigned char> specialization.
- struct std::numeric_limits< unsigned int >

  numeric_limits<unsigned int> specialization.
- struct std::numeric_limits< unsigned long >

  numeric_limits<unsigned long> specialization.
- struct std::numeric_limits< unsigned long long >

  numeric_limits<unsigned long long> specialization.
- struct std::numeric_limits< unsigned short >

  numeric_limits<unsigned short> specialization.
- struct std::numeric_limits< wchar_t >

  numeric_limits<wchar_t> specialization.

Namespaces

- namespace std

Macros

- #define **__glibcxx_digits**(T)
- #define **__glibcxx_digits10**(T)
- #define **__glibcxx_double_has_denorm_loss**
- #define **__glibcxx_double_tinyness_before**
- #define **__glibcxx_double_traps**
- #define **__glibcxx_float_has_denorm_loss**
- #define **__glibcxx_float_tinyness_before**

- #define **__glibcxx_float_traps**
- #define **__glibcxx_integral_traps**
- #define **__glibcxx_long_double_has_denorm_loss**
- #define **__glibcxx_long_double_tinyness_before**
- #define **__glibcxx_long_double_traps**
- #define **__glibcxx_max**(T)
- #define **__glibcxx_max_digits10**(T)
- #define **__glibcxx_min**(T)
- #define **__glibcxx_signed**(T)
- #define **_GLIBCXX_NUMERIC_LIMITS**

Enumerations

- enum std::float_denorm_style { std::denorm_indeterminate, std::denorm_absent, std::denorm_present }
- enum std::float_round_style { **round_indeterminate**, std::round_toward_zero, std::round_to_nearest, std::round_toward_infinity, std::round_toward_neg_infinity }

5.303.1 Detailed Description

This is a Standard C++ Library header.

Definition in file limits.

5.304 linear_probe_fn_imp.hpp File Reference

5.304.1 Detailed Description

Contains a probe policy implementation

Definition in file linear_probe_fn_imp.hpp.

5.305 list File Reference

Macros

- #define **_GLIBCXX_LIST**

5.305.1 Detailed Description

This is a Standard C++ Library header.

Definition in file list.

5.306 list File Reference

Classes

- class std::__debug::list< _Tp, _Allocator >

  Class std::list with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_LIST**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator!=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator>** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator>=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__debug::swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)

5.306.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/list.

5.307 list File Reference

Classes

- class std::__profile::list< _Tp, _Allocator >

  List wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Macros

- #define **_GLIBCXX_PROFILE_LIST**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator!=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator<** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator<=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator>** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator>=** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__profile::swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs)

5.307.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/list.

5.308 list.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_LIST_TCC**

5.308.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <list>.

Definition in file list.tcc.

5.309 list_partition.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter >
  void __gnu_parallel::__shrink (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length)

- template<typename _IIter >
  void __gnu_parallel::__shrink_and_double (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)
- template<typename _IIter , typename _FunctorType >
  size_t __gnu_parallel::list_partition (const _IIter __begin, const _IIter __end, _IIter ∗__starts, size_t ∗__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)

5.309.1 Detailed Description

_Functionality to split __sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file list_partition.h.

5.310 list_update_policy.hpp File Reference

Classes

- class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
- class __gnu_pbds::lu_move_to_front_policy< _Alloc >

Namespaces

- namespace __gnu_pbds

5.310.1 Detailed Description

Contains policies for list update containers.

Definition in file list_update_policy.hpp.

5.311 locale File Reference

Macros

- #define **_GLIBCXX_LOCALE**

5.311.1 Detailed Description

This is a Standard C++ Library header.

Definition in file locale.

5.312 locale_classes.h File Reference

Classes

- class std::collate< _CharT >

  Facet for localized string comparison.
- class std::collate_byname< _CharT >

class collate_byname [22.2.4.2].

- class std::locale

  Container class for localization functionality.
  The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.
- class std::locale::facet

  Localization functionality base class.
  The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.
- class std::locale::id

  Facet ID class.
  The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.

Namespaces

- namespace std

5.312.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file locale_classes.h.

5.313 locale_classes.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_LOCALE_CLASSES_TCC**

Functions

- template<typename _Facet >
  bool std::has_facet (const locale &__loc) throw ()
- template<typename _Facet >
  const _Facet & std::use_facet (const locale &__loc)

5.313.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file locale_classes.tcc.

5.314 locale_facets.h File Reference

Classes

- class std::__ctype_abstract_base< _CharT >

  Common base for ctype facet.
- class std::ctype< _CharT >

  Primary class template ctype facet.
  This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.
- class std::ctype< char >

  The ctype<char> specialization.
  This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.
- class std::ctype< wchar_t >

  The ctype<wchar_t> specialization.
  This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
- class std::ctype_byname< _CharT >

  class ctype_byname [22.2.1.2].
- class std::ctype_byname< char >

  22.2.1.4 Class ctype_byname specializations.
- class std::num_get< _CharT, _InIter >

  Primary class template num_get.
  This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
- class std::num_put< _CharT, _OutIter >

  Primary class template num_put.
  This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.
- class std::numpunct< _CharT >

  Primary class template numpunct.
  This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
- class std::numpunct_byname< _CharT >

  class numpunct_byname [22.2.3.2].

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_NUM_FACETS**

Functions

- template<typename _CharT >
  _CharT ∗ **std::__add_grouping** (_CharT ∗__s, _CharT __sep, const char ∗__gbeg, size_t __gsize, const _CharT ∗__first, const _CharT ∗__last)

- template<typename _Tp >
  void **std::__convert_to_v** (const char ∗, _Tp &, ios_base::iostate &, const __c_locale &) throw ()
- template<>
  void **std::__convert_to_v** (const char ∗, float &, ios_base::iostate &, const __c_locale &) throw ()
- template<>
  void **std::__convert_to_v** (const char ∗, double &, ios_base::iostate &, const __c_locale &) throw ()
- template<>
  void **std::__convert_to_v** (const char ∗, long double &, ios_base::iostate &, const __c_locale &) throw ()
- template<typename _CharT >
  ostreambuf_iterator< _CharT > **std::__write** (ostreambuf_iterator< _CharT > __s, const _CharT ∗__ws, int __len)
- template<typename _CharT , typename _OutIter >
  _OutIter **std::__write** (_OutIter __s, const _CharT ∗__ws, int __len)
- template<typename _CharT >
  bool std::isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::islower (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isprint (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isspace (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isupper (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isxdigit (_CharT __c, const locale &__loc)
- template<typename _CharT >
  _CharT std::tolower (_CharT __c, const locale &__loc)
- template<typename _CharT >
  _CharT std::toupper (_CharT __c, const locale &__loc)

5.314.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file locale_facets.h.

5.315 locale_facets.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_LOCALE_FACETS_TCC**

Functions

- template<typename _CharT >
  _CharT ∗ **std::__add_grouping** (_CharT ∗__s, _CharT __sep, const char ∗__gbeg, size_t __gsize, const _CharT ∗__first, const _CharT ∗__last)
- template<typename _CharT , typename _ValueT >
  int **std::__int_to_char** (_CharT ∗__bufend, _ValueT __v, const _CharT ∗__lit, ios_base::fmtflags __flags, bool __dec)
- bool **std::__verify_grouping** (const char ∗__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()

5.315.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file locale_facets.tcc.

5.316 locale_facets_nonio.h File Reference

Classes

- class std::messages< _CharT >

  Primary class template messages.
  This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.
- struct std::messages_base

  Messages facet base class providing catalog typedef.
- class std::messages_byname< _CharT >

  class messages_byname [22.2.7.2].
- class std::money_base

  Money format ordering data.
  This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.
- class std::money_get< _CharT, _InIter >

  Primary class template money_get.
  This facet encapsulates the code to parse and return a monetary amount from a string.
- class std::money_put< _CharT, _OutIter >

  Primary class template money_put.
  This facet encapsulates the code to format and output a monetary amount.
- class std::moneypunct< _CharT, _Intl >

  Primary class template moneypunct.
  This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.
- class std::moneypunct_byname< _CharT, _Intl >

  class moneypunct_byname [22.2.6.4].

- class std::time_base

  Time format ordering data.
  This class provides an enum representing different orderings of time: day, month, and year.
- class std::time_get< _CharT, _InIter >

  Primary class template time_get.
  This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.
- class std::time_get_byname< _CharT, _InIter >

  class time_get_byname [22.2.5.2].
- class std::time_put< _CharT, _OutIter >

  Primary class template time_put.
  This facet encapsulates the code to format and output dates and times according to formats used by strftime().
- class std::time_put_byname< _CharT, _OutIter >

  class time_put_byname [22.2.5.4].

Namespaces

- namespace std

5.316.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file locale_facets_nonio.h.

5.317 locale_facets_nonio.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_LOCALE_FACETS_NONIO_TCC**

5.317.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file locale_facets_nonio.tcc.

5.318 localefwd.h File Reference

Classes

- class std::codecvt< _InternT, _ExternT, _StateT >

Primary class template codecvt.
NB: Generic, mostly useless implementation.

- class std::codecvt_byname< _InternT, _ExternT, _StateT >

  class codecvt_byname [22.2.1.6].

- class std::collate< _CharT >

  Facet for localized string comparison.

- class std::collate_byname< _CharT >

  class collate_byname [22.2.4.2].

- class std::ctype< _CharT >

  Primary class template ctype facet.
  This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.

- class std::ctype_byname< _CharT >

  class ctype_byname [22.2.1.2].

- class std::messages< _CharT >

  Primary class template messages.
  This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

- class std::messages_byname< _CharT >

  class messages_byname [22.2.7.2].

- class std::money_get< _CharT, _InIter >

  Primary class template money_get.
  This facet encapsulates the code to parse and return a monetary amount from a string.

- class std::money_put< _CharT, _OutIter >

  Primary class template money_put.
  This facet encapsulates the code to format and output a monetary amount.

- class std::moneypunct< _CharT, _Intl >

  Primary class template moneypunct.
  This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

- class std::moneypunct_byname< _CharT, _Intl >

  class moneypunct_byname [22.2.6.4].

- class std::num_get< _CharT, _InIter >

  Primary class template num_get.
  This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

- class std::num_put< _CharT, _OutIter >

  Primary class template num_put.
  This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

- class std::numpunct< _CharT >

  Primary class template numpunct.
  This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

- class std::numpunct_byname< _CharT >

  class numpunct_byname [22.2.3.2].

- class std::time_get< _CharT, _InIter >

  Primary class template time_get.
  This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

- class std::time_get_byname< _CharT, _InIter >

class time_get_byname [22.2.5.2].

- class std::time_put< _CharT, _OutIter >

  Primary class template time_put.
  This facet encapsulates the code to format and output dates and times according to formats used by strftime().
- class std::time_put_byname< _CharT, _OutIter >

  class time_put_byname [22.2.5.4].

Namespaces

- namespace std

Functions

- template<typename _Facet >
  bool std::has_facet (const locale &__loc) throw ()
- template<typename _CharT >
  bool std::isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::islower (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isprint (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isspace (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isupper (_CharT __c, const locale &__loc)
- template<typename _CharT >
  bool std::isxdigit (_CharT __c, const locale &__loc)
- template<typename _CharT >
  _CharT std::tolower (_CharT __c, const locale &__loc)
- template<typename _CharT >
  _CharT std::toupper (_CharT __c, const locale &__loc)
- template<typename _Facet >
  const _Facet & std::use_facet (const locale &__loc)

5.318.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file localefwd.h.

5.319 losertree.h File Reference

Classes

- class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >

  Stable _LoserTree variant.
- class __gnu_parallel::_LoserTree< false, _Tp, _Compare >

  Unstable _LoserTree variant.
- class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >

  Guarded loser/tournament tree.
- struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser

  Internal representation of a _LoserTree element.
- class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >

  Stable _LoserTree implementation.
- class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >

  Unstable _LoserTree implementation.
- class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >

  Base class of _Loser Tree implementation using pointers.
- struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser

  Internal representation of _LoserTree __elements.
- class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >

  Stable unguarded _LoserTree variant storing pointers.
- class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >

  Unstable unguarded _LoserTree variant storing pointers.
- class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >

  Unguarded loser tree, keeping only pointers to the elements in the tree structure.
- class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >

  Stable implementation of unguarded _LoserTree.
- class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >

  Non-Stable implementation of unguarded _LoserTree.
- class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >

  Base class for unguarded _LoserTree implementation.

Namespaces

- namespace __gnu_parallel

5.319.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file losertree.h.

5.320 lu_counter_metadata.hpp File Reference

Classes

- class __gnu_pbds::detail::lu_counter_metadata< Size_Type >

  A list-update metadata type that moves elements to the front of the list based on the counter algorithm.
- class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >

  Base class for list-update counter policy.

Namespaces

- namespace __gnu_pbds

5.320.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

Definition in file lu_counter_metadata.hpp.

5.321 lu_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >

  list-based (with updates) associative container. Skip to the lu, my darling.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_LU_NAME**
- #define **PB_DS_LU_TRAITS_BASE**

5.321.1 Detailed Description

Contains a list update map.

Definition in file lu_map_.hpp.

5.322 macros.h File Reference

Macros

- #define **__glibcxx_check_bucket_index**(_N)
- #define **__glibcxx_check_equal_allocs**(_Other)
- #define __glibcxx_check_erase(_Position)
- #define __glibcxx_check_erase_after(_Position)
- #define __glibcxx_check_erase_range(_First, _Last)
- #define __glibcxx_check_erase_range_after(_First, _Last)
- #define **__glibcxx_check_heap**(_First, _Last)
- #define __glibcxx_check_heap_pred(_First, _Last, _Pred)
- #define __glibcxx_check_insert(_Position)
- #define __glibcxx_check_insert_after(_Position)
- #define __glibcxx_check_insert_range(_Position, _First, _Last)
- #define __glibcxx_check_insert_range_after(_Position, _First, _Last)
- #define **__glibcxx_check_max_load_factor**(_F)
- #define **__glibcxx_check_non_empty_range**(_First, _Last)
- #define **__glibcxx_check_nonempty**()
- #define __glibcxx_check_partitioned_lower(_First, _Last, _Value)
- #define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)
- #define **__glibcxx_check_partitioned_upper**(_First, _Last, _Value)
- #define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)
- #define **__glibcxx_check_self_move_assign**(_Other)
- #define **__glibcxx_check_sorted**(_First, _Last)
- #define __glibcxx_check_sorted_pred(_First, _Last, _Pred)
- #define **__glibcxx_check_sorted_set**(_First1, _Last1, _First2)
- #define **__glibcxx_check_sorted_set_pred**(_First1, _Last1, _First2, _Pred)
- #define **__glibcxx_check_string**(_String)
- #define **__glibcxx_check_string_len**(_String, _Len)
- #define **__glibcxx_check_subscript**(_N)
- #define **__glibcxx_check_valid_range**(_First, _Last)
- #define **_GLIBCXX_DEBUG_VERIFY**(_Condition, _ErrorMessage)
- #define _GLIBCXX_DEBUG_VERIFY_AT(_Condition, _ErrorMessage, _File, _Line)

5.322.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file macros.h.

5.322.2 Macro Definition Documentation

5.322.2.1 #define __glibcxx_check_erase(*_Position*)

Verify that we can erase the element referenced by the iterator _Position. We can erase the element if the _Position iterator is dereferenceable and references this sequence.

Definition at line 135 of file macros.h.

5.322.2.2 #define __glibcxx_check_erase_after(*_Position*)

Verify that we can erase the element after the iterator _Position. We can erase the element if the _Position iterator is before a dereferenceable one and references this sequence.

Definition at line 149 of file macros.h.

5.322.2.3 #define __glibcxx_check_erase_range(*_First, _Last*)

Verify that we can erase the elements in the iterator range [_First, _Last). We can erase the elements if [_First, _Last) is a valid iterator range within this sequence.

Definition at line 163 of file macros.h.

5.322.2.4 #define __glibcxx_check_erase_range_after(*_First, _Last*)

Verify that we can erase the elements in the iterator range (_First, _Last). We can erase the elements if (_First, _Last) is a valid iterator range within this sequence.

Definition at line 175 of file macros.h.

5.322.2.5 #define __glibcxx_check_heap_pred(*_First, _Last, _Pred*)

Verify that the iterator range [_First, _Last) is a heap w.r.t. the predicate _Pred.

Definition at line 314 of file macros.h.

5.322.2.6 #define __glibcxx_check_insert(*_Position*)

Verify that we can insert into ∗this with the iterator _Position. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a_Safe_sequence and the iterator is a _Safe_iterator.

Definition at line 73 of file macros.h.

5.322.2.7 #define __glibcxx_check_insert_after(*_Position*)

Verify that we can insert into ∗this after the iterator _Position. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a_Safe_sequence and the iterator is a _Safe_iterator.

Definition at line 90 of file macros.h.

5.322.2.8 #define __glibcxx_check_insert_range(*_Position, _First, _Last*)

Verify that we can insert the values in the iterator range [_First, _Last) into ∗this with the iterator _Position. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [_First, Last) is a valid (possibly empty) range. Note that this macro is only valid when the container is a _Safe_sequence and the iterator is a _Safe_iterator.

Todo We would like to be able to check for noninterference of _Position and the range [_First, _Last), but that can't (in general) be done.

Definition at line 110 of file macros.h.

5.322.2.9 #define __glibcxx_check_insert_range_after(*_Position,* *_First,* *_Last*)

Verify that we can insert the values in the iterator range [_First, _Last) into ∗this after the iterator _Position. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [_First, Last) is a valid (possibly empty) range. Note that this macro is only valid when the container is a _Safe_sequence and the iterator is a _Safe_iterator.

Todo We would like to be able to check for noninterference of _Position and the range [_First, _Last), but that can't (in general) be done.

Definition at line 127 of file macros.h.

5.322.2.10 #define __glibcxx_check_partitioned_lower(*_First,* *_Last,* *_Value*)

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value.

Definition at line 262 of file macros.h.

5.322.2.11 #define __glibcxx_check_partitioned_lower_pred(*_First,* *_Last,* *_Value,* *_Pred*)

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value and predicate _Pred.

Definition at line 282 of file macros.h.

5.322.2.12 #define __glibcxx_check_partitioned_upper_pred(*_First,* *_Last,* *_Value,* *_Pred*)

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value and predicate _Pred.

Definition at line 294 of file macros.h.

5.322.2.13 #define __glibcxx_check_sorted_pred(*_First,* *_Last,* *_Pred*)

Verify that the iterator range [_First, _Last) is sorted by the predicate _Pred.

Definition at line 233 of file macros.h.

5.322.2.14 #define _GLIBCXX_DEBUG_VERIFY_AT(*_Condition,* *_ErrorMessage,* *_File,* *_Line*)

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 41 of file macros.h.

5.323 malloc_allocator.h File Reference

Classes

- class __gnu_cxx::malloc_allocator< _Tp >

  An allocator that uses malloc.
  This is precisely the allocator defined in the C++ Standard.

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp >
  bool **__gnu_cxx::operator!=** (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)
- template<typename _Tp >
  bool **__gnu_cxx::operator==** (const malloc_allocator< _Tp > &, const malloc_allocator< _Tp > &)

5.323.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file malloc_allocator.h.

5.324 map File Reference

Macros

- #define **_GLIBCXX_MAP**

5.324.1 Detailed Description

This is a Standard C++ Library header.

Definition in file map.

5.325 map File Reference

Macros

- #define **_GLIBCXX_DEBUG_MAP**

5.325.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/map.

5.326 map File Reference

Macros

- #define **_GLIBCXX_PROFILE_MAP**

5.326.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/map.

5.327 map.h File Reference

Classes

- class std::__debug::map< _Key, _Tp, _Compare, _Allocator >

  Class std::map with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator!=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **std::__debug::swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)

5.327.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/map.h.

5.328 map.h File Reference

Classes

- class std::__profile::map< _Key, _Tp, _Compare, _Allocator >

  Class std::map wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Functions

- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator!=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>=** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **std::__profile::swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs)

5.328.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/map.h.

5.329 mask_array.h File Reference

Classes

- class std::mask_array< _Tp >

  Reference to selected subset of an array.

Namespaces

- namespace std

Macros

- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)

5.329.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file mask_array.h.

5.330 mask_based_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >

  Range hashing policy.

Namespaces

- namespace __gnu_pbds

5.330.1 Detailed Description

Contains a range hashing policy base.

Definition in file mask_based_range_hashing.hpp.

5.331 memory File Reference

Macros

- #define **_GLIBCXX_MEMORY**

5.331.1 Detailed Description

This is a Standard C++ Library header.

Definition in file memory.

5.332 memory File Reference

Classes

- struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_MEMORY**

Functions

- template< typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)
- template< typename _RandomAccessIter , typename _Size , typename _ForwardIter >
  pair< _RandomAccessIter,
  _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_RandomAccessIter __first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)
- template< typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result)
- template< typename _InputIter , typename _Size , typename _ForwardIter , typename _Allocator >
  pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)
- template< typename _InputIter , typename _Size , typename _ForwardIter , typename _Tp >
  pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)
- template< typename _InputIter , typename _Size , typename _ForwardIter >
  pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)

5.332.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file ext/memory.

5.333 merge.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template< typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template< typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template< typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template< typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare >
  _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template< typename _RAIter1 , typename _RAIter3 , typename _Compare >
  _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)

5.333.1 Detailed Description

Parallel implementation of std::merge(). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file merge.h.

5.334 messages_members.h File Reference

Namespaces

- namespace std

5.334.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file messages_members.h.

5.335 mod_based_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >

  Mod based range hashing.

Namespaces

- namespace __gnu_pbds

5.335.1 Detailed Description

Contains a range hashing policy base.

Definition in file mod_based_range_hashing.hpp.

5.336 move.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FORWARD**(_Tp, __val)
- #define **_GLIBCXX_MOVE**(__val)

Functions

- template<typename _Tp >
  _Tp ∗ std::__addressof (_Tp &__r) noexcept
- template<typename _Tp >
  _Tp ∗ std::addressof (_Tp &__r) noexcept
- template<typename _Tp >
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename _Tp >
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename _Tp >
  constexpr
  std::remove_reference< _Tp >
  ::type && std::move (_Tp &&__t) noexcept
- template<typename _Tp >
  constexpr conditional
  < __move_if_noexcept_cond< _Tp >
  ::value, const _Tp &, _Tp && >
  ::type std::move_if_noexcept (_Tp &__x) noexcept
- template<typename _Tp >
  void std::swap (_Tp &__a, _Tp &__b) noexcept(__and_< is_nothrow_move_constructible< _Tp >
- template<typename _Tp , size_t _Nm>
  void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(noexcept(swap(∗__a

Variables

- **std::__a**
- void
  is_nothrow_move_assignable
  < _Tp >::value _Tp **std::__tmp**

5.336.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

Definition in file move.h.

5.337 mt_allocator.h File Reference

Classes

- struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >

  Policy for shared __pool objects.
- class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

  This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a global one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the global list). Further details: `http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt12ch32.html`.
- class __gnu_cxx::__mt_alloc_base< _Tp >

  Base class for _Tp dependent member functions.
- struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >

Policy for individual __pool objects.
- class __gnu_cxx::__pool< false >

  Specialization for single thread.
- class __gnu_cxx::__pool< true >

  Specialization for thread enabled, via gthreads.h.
- struct __gnu_cxx::__pool_base

  Base class for pool object.
- class __pool< _Thread >

  Data describing the underlying memory pool, parameterized on threading support.

Namespaces

- namespace __gnu_cxx

Macros

- #define **__thread_default**

Typedefs

- typedef void(∗ **__gnu_cxx::__destroy_handler**)(void ∗)

Functions

- template<typename _Tp , typename _Poolp >
  bool **__gnu_cxx::operator!=** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)
- template<typename _Tp , typename _Poolp >
  bool **__gnu_cxx::operator==** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)

5.337.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file mt_allocator.h.

5.338 multimap.h File Reference

Classes

- class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >

  Class std::multimap with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator!=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **std::__debug::swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)

5.338.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/multimap.h.

5.339 multimap.h File Reference

Classes

- class std::__profile::multimap< _Key, _Tp, _Compare, _Allocator >

  Class std::multimap wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Functions

- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator!=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)

- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>=** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Tp , typename _Compare , typename _Allocator >
  void **std::__profile::swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)

5.339.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/multimap.h.

5.340 multiseq_selection.h File Reference

Classes

- class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >

  Compare __a pair of types lexicographically, ascending.
- class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >

  Compare __a pair of types lexicographically, descending.

Namespaces

- namespace __gnu_parallel

Macros

- #define **__S**(__i)
- #define **__S**(__i)

Functions

- template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare >
  void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())
- template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare >
  _Tp __gnu_parallel::multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())

5.340.1 Detailed Description

Functions to find elements of a certain global __rank in multiple sorted sequences. Also serves for splitting such sequence sets. The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171–177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file multiseq_selection.h.

5.341 multiset.h File Reference

Classes

- class std::__debug::multiset< _Key, _Compare, _Allocator >

  Class std::multiset with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator!=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **std::__debug::swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)

5.341.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/multiset.h.

5.342 multiset.h File Reference

Classes

- class std::__profile::multiset< _Key, _Compare, _Allocator >

  Class std::multiset wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Functions

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator!=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>=** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **std::__profile::swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y)

5.342.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/multiset.h.

5.343 multiway_merge.h File Reference

Classes

- struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for 3-way merging with __sentinels turned off.
- struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned on.

- struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for 4-way merging with __sentinels turned off.
- struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for 4-way merging with __sentinels turned on.
- struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for k-way merging with __sentinels turned on.
- struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

  Switch for k-way merging with __sentinels turned off.
- class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >

  _Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.
- struct __gnu_parallel::_LoserTreeTraits< _Tp >

  Traits for determining whether the loser tree should use pointers or copies.
- struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >

  Stable sorting functor.
- struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >

  Non-__stable sorting functor.

Namespaces

- namespace __gnu_parallel

Macros

- #define **_GLIBCXX_PARALLEL_DECISION**(__a, __b, __c, __d)
- #define _GLIBCXX_PARALLEL_LENGTH(__s)
- #define **_GLIBCXX_PARALLEL_MERGE_3_CASE**(__a, __b, __c, __c0, __c1)
- #define **_GLIBCXX_PARALLEL_MERGE_4_CASE**(__a, __b, __c, __d,__c0, __c1, __c2)

Functions

- template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
  _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __gnu_parallel::__sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)

- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<template< typename RAI, typename C > class iterator, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
  void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > ∗__pieces)
- template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<typename UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Compare >
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
  void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > ∗__pieces)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)

- template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp , typename _Splitter , typename _Compare >
_RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _Compare >
_RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)

5.343.1 Detailed Description

Implementation of sequential and parallel multiway merge. Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file multiway_merge.h.

5.343.2 Macro Definition Documentation

5.343.2.1 #define _GLIBCXX_PARALLEL_LENGTH(__s)

Length of a sequence described by a pair of iterators.

Definition at line 54 of file multiway_merge.h.

Referenced by __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::multiway_merge_exact_splitting(), __gnu_parallel::multiway_merge_loser_tree(), __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiway_merge().

5.344 multiway_mergesort.h File Reference

Classes

- struct __gnu_parallel::_Piece< _DifferenceTp >

  Subsequence description.
- struct __gnu_parallel::_PMWMSSortingData< _RAIter >

  Data accessed by all threads.
- struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >

  Split consistently.
- struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >

  Split by sampling.
- struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

  Split by exact splitting.

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter , typename _DifferenceTp >
  void __gnu_parallel::__determine_samples (_PMWMSSortingData< _RAIter > *__sd, _DifferenceTp __num_samples)
- template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
  void __gnu_parallel::parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
  void __gnu_parallel::parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)

5.344.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file multiway_mergesort.h.

5.345 mutex File Reference

Classes

- struct std::adopt_lock_t

Assume the calling thread has already obtained mutex ownership and manage it.

- struct std::defer_lock_t

  Do not acquire ownership of the mutex.

- class std::lock_guard< _Mutex >

  Scoped lock idiom.

- class std::mutex

  mutex

- struct std::once_flag

  once_flag

- class std::recursive_mutex

  recursive_mutex

- struct std::try_to_lock_t

  Try to acquire ownership of the mutex without blocking.

- class std::unique_lock< _Mutex >

  unique_lock

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MUTEX**

Functions

- mutex & **std::__get_once_mutex** ()
- void **std::__once_proxy** (void)
- void **std::__set_once_functor_lock_ptr** (unique_lock< mutex > ∗)
- template<typename _Lock >
  unique_lock< _Lock > **std::__try_to_lock** (_Lock &__l)
- template<typename _Callable , typename... _Args>
  void std::call_once (once_flag &__once, _Callable &&__f, _Args &&...__args)
- template<typename _L1 , typename _L2 , typename... _L3>
  void std::lock (_L1 &__l1, _L2 &__l2, _L3 &...__l3)
- template<typename _Mutex >
  void std::swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept
- template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
  int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &...__l3)

Variables

- function< void()> **std::__once_functor**
- constexpr adopt_lock_t **std::adopt_lock**
- constexpr defer_lock_t **std::defer_lock**
- constexpr try_to_lock_t **std::try_to_lock**

5.345.1 Detailed Description

This is a Standard C++ Library header.

Definition in file mutex.

5.346 nested_exception.h File Reference

Classes

- class std::nested_exception

  Exception class with exception_ptr data member.

Namespaces

- namespace std

Functions

- template<typename _Ex >
  const nested_exception ∗ **std::__get_nested_exception** (const _Ex &__ex)
- template<typename _Ex >
  void **std::__throw_with_nested** (_Ex &&, const nested_exception ∗=0) __attribute__((__noreturn__))
- template<typename _Ex >
  void **std::__throw_with_nested** (_Ex &&,...) __attribute__((__noreturn__))
- template<typename _Ex >
  void std::rethrow_if_nested (const _Ex &__ex)
- void std::rethrow_if_nested (const nested_exception &__ex)
- template<typename _Ex >
  void std::throw_with_nested (_Ex __ex)

5.346.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

Definition in file nested_exception.h.

5.347 new File Reference

Classes

- class std::bad_alloc

  Exception possibly thrown by `new`.
  `bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Namespaces

- namespace std

Typedefs

- typedef void(∗ std::new_handler)()

Functions

- new_handler std::set_new_handler (new_handler) throw ()

- void ∗ operator new (std::size_t) __attribute__((__externally_visible__))
- void ∗ operator new[] (std::size_t) __attribute__((__externally_visible__))
- void operator delete (void ∗) noexcept __attribute__((__externally_visible__))
- void operator delete[] (void ∗) noexcept __attribute__((__externally_visible__))
- void ∗ operator new (std::size_t, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void ∗ operator new[] (std::size_t, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void operator delete (void ∗, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void operator delete[] (void ∗, const std::nothrow_t &) noexcept __attribute__((__externally_visible__))
- void ∗ operator new (std::size_t, void ∗__p) noexcept
- void ∗ operator new[] (std::size_t, void ∗__p) noexcept
- void operator delete (void ∗, void ∗) noexcept
- void operator delete[] (void ∗, void ∗) noexcept

Variables

- const nothrow_t **std::nothrow**

5.347.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see `http://gcc.gnu.org/onlinedocs/libstdc++/18_support/howto.html#4` for more.

Definition in file new.

5.347.2 Function Documentation

5.347.2.1 void operator delete (void ∗) [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.2 void operator delete (void *, const std::nothrow_t &) `[noexcept]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.3 void operator delete (void *, void *) `[inline]`, `[noexcept]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 115 of file new.

5.347.2.4 void operator delete[] (void *) `[noexcept]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.5 void operator delete[] (void *, const std::nothrow_t &) `[noexcept]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.6 void operator delete[] (void *, void *) [inline], [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 116 of file new.

5.347.2.7 void* operator new (std::size_t)

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.8 void* operator new (std::size_t, const std::nothrow_t &) [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.9 void* operator new (std::size_t, void * __p) [inline], [noexcept]

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 109 of file new.

5.347.2.10 void∗ operator new[] (std::size_t)

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.11 void∗ operator new[] (std::size_t , const std::nothrow_t &) `[noexcept]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

5.347.2.12 void∗ operator new[] (std::size_t , void ∗ *__p*) `[inline]`, `[noexcept]`

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 111 of file new.

5.348 new_allocator.h File Reference

Classes

- class __gnu_cxx::new_allocator< _Tp >

  An allocator that uses global new, as per [20.4].
  This is precisely the allocator defined in the C++ Standard.

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp >
  bool **__gnu_cxx::operator!=** (const new_allocator< _Tp > &, const new_allocator< _Tp > &)
- template<typename _Tp >
  bool **__gnu_cxx::operator==** (const new_allocator< _Tp > &, const new_allocator< _Tp > &)

5.348.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file new_allocator.h.

5.349 node.hpp File Reference

Classes

- struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >

  Node.

Namespaces

- namespace __gnu_pbds

5.349.1 Detailed Description

Contains an implementation struct for this type of heap's node.

Definition in file left_child_next_sibling_heap_/node.hpp.

5.350 node.hpp File Reference

Classes

- struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >

  Node for Red-Black trees.

Namespaces

- namespace __gnu_pbds

5.350.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/node.hpp.

5.351 node.hpp File Reference

Classes

- struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >

  Node for splay tree.

Namespaces

- namespace __gnu_pbds

5.351.1 Detailed Description

Contains an implementation struct for splay_tree_'s node.

Definition in file splay_tree_/node.hpp.

5.352 node_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >

  Const node iterator.
- class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

  Node iterator.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC**
- #define **PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC**

5.352.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/node_iterators.hpp.

5.353 node_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >

  Const node reference.
- class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >

  Node reference.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC**
- #define **PB_DS_OV_TREE_NODE_ITERATOR_C_DEC**

5.353.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/node_iterators.hpp.

5.354 node_metadata_selector.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
  Specialization, false.
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
  Specialization, true.
- struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
  Tree node metadata dispatch.
- struct tree_metadata_helper< Node_Update, _BTp >
  Tree metadata helper.

Namespaces

- namespace __gnu_pbds

5.354.1 Detailed Description

Contains an implementation class for trees.

Definition in file tree_policy/node_metadata_selector.hpp.

5.355 node_metadata_selector.hpp File Reference

Classes

- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
  Specialization, false.
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
  Specialization, true.
- struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
  Trie node metadata dispatch.
- struct trie_metadata_helper< Node_Update, _BTp >
  Trie metadata helper.

Namespaces

- namespace __gnu_pbds

5.355.1 Detailed Description

Contains an implementation class for tries.

Definition in file trie_policy/node_metadata_selector.hpp.

5.356 null_node_metadata.hpp File Reference

Classes

- struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >

  Constant node iterator.

Namespaces

- namespace __gnu_pbds

5.356.1 Detailed Description

Contains an implementation class for tree-like classes.

Definition in file null_node_metadata.hpp.

5.357 numeric File Reference

Macros

- #define **_GLIBCXX_NUMERIC**

5.357.1 Detailed Description

This is a Standard C++ Library header.

Definition in file numeric.

5.358 numeric File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_NUMERIC**

Functions

- template<typename _Tp , typename _Integer , typename _MonoidOperation >
  _Tp **__gnu_cxx::__power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _Tp , typename _Integer >
  _Tp **__gnu_cxx::__power** (_Tp __x, _Integer __n)
- template<typename _Tp , typename _Integer , typename _MonoidOperation >
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _Tp , typename _Integer >
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n)

5.358.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file ext/numeric.

5.359 numeric File Reference

Namespaces

- namespace std
- namespace std::__parallel

Macros

- #define **_GLIBCXX_PARALLEL_NUMERIC_H**

Functions

- template<typename _IIter , typename _Tp , typename _IteratorTag >
  _Tp **std::__parallel::__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _IteratorTag)
- template<typename _IIter , typename _Tp , typename _BinaryOperation , typename _IteratorTag >
  _Tp **std::__parallel::__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)
- template<typename __RAIter , typename _Tp , typename _BinaryOperation >
  _Tp **std::__parallel::__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >
  _OutputIterator **std::__parallel::__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **std::__parallel::__inner_product_switch** (_RAIter1 __first1, _RAIter1 __last1, _RAIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag=__gnu_parallel::parallel_unbalanced)

- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 , typename _IteratorTag1 , typename _IteratorTag2 >
  _Tp **std::__parallel::__inner_product_switch** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation , typename _IteratorTag1 , typename _IteratorTag2 >
  _OutputIterator **std::__parallel::__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter , typename _Tp >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter , typename _Tp >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _Tp , typename _BinaryOperation >
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp , typename _BinaryOperation >
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _Tp , typename _BinaryOperation >
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)

- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **std::__parallel::inner_product** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)

5.359.1 Detailed Description

Parallel STL function calls corresponding to stl_numeric.h. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file parallel/numeric.

5.360 numeric_traits.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **__glibcxx_digits**(_Tp)
- #define **__glibcxx_digits10**(_Tp)
- #define **__glibcxx_floating**(_Tp, _Fval, _Dval, _LDval)
- #define **__glibcxx_max**(_Tp)
- #define **__glibcxx_max_digits10**(_Tp)
- #define **__glibcxx_max_exponent10**(_Tp)
- #define **__glibcxx_min**(_Tp)
- #define **__glibcxx_signed**(_Tp)

5.360.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file numeric_traits.h.

5.361 numericfwd.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _IIter , typename _Tp , typename _Tag >
  _Tp **std::__parallel::__accumulate_switch** (_IIter, _IIter, _Tp, _Tag)
- template<typename _IIter , typename _Tp , typename _BinaryOper , typename _Tag >
  _Tp **std::__parallel::__accumulate_switch** (_IIter, _IIter, _Tp, _BinaryOper, _Tag)
- template<typename _RAIter , typename _Tp , typename _BinaryOper >
  _Tp **std::__parallel::__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter , typename _OIter , typename _BinaryOper , typename _Tag1 , typename _Tag2 >
  _OIter **std::__parallel::__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _RAIter1 , typename _RAIter2 , typename _Tp , typename BinaryFunction1 , typename BinaryFunction2 >
  _Tp **std::__parallel::__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, BinaryFunction1, BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 , typename _Tag1 , typename _Tag2 >
  _Tp **std::__parallel::__inner_product_switch** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)
- template<typename _IIter , typename _OIter , typename _BinaryOper , typename _Tag1 , typename _Tag2 >
  _OIter **std::__parallel::__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter , typename _Tp >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter , typename _Tp >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _Tp , typename _BinaryOper >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper)
- template<typename _IIter , typename _Tp , typename _BinaryOper >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _Tp , typename _BinaryOper >
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)

- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Tp >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename _BinaryFunction1 , typename _BinaryFunction2 >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter1 , typename _IIter2 , typename _Tp , typename BinaryFunction1 , typename BinaryFunction2 >
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, BinaryFunction1, BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter , typename _OIter >
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter __result)
- template<typename _IIter , typename _OIter , typename _BinaryOper >
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, _BinaryOper)

5.361.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file numericfwd.h.

5.362 omp_loop.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

5.362.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file omp_loop.h.

5.363 omp_loop_static.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

5.363.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file omp_loop_static.h.

5.364 opt_random.h File Reference

Namespaces

- namespace std

5.364.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

Definition in file bits/opt_random.h.

5.365 order_statistics_imp.hpp File Reference

5.365.1 Detailed Description

Contains forward declarations for order_statistics_key

Definition in file tree_policy/order_statistics_imp.hpp.

5.366 order_statistics_imp.hpp File Reference

5.366.1 Detailed Description

Contains forward declarations for order_statistics_key

Definition in file trie_policy/order_statistics_imp.hpp.

5.367 os_defines.h File Reference

Macros

- #define **__NO_CTYPE**

5.367.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

Definition in file os_defines.h.

5.368 ostream File Reference

Classes

- class std::basic_ostream< _CharT, _Traits >

  Template class basic_ostream.
- class std::basic_ostream< _CharT, _Traits >::sentry

  Performs setup work for output streams.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_OSTREAM**

Functions

- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::endl (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::ends (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::flush (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT , typename _Traits , typename _Tp >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &&__os, const _Tp &__x)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)
- template<class _Traits >
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)
- template<class _Traits >
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)

- template<class _Traits >
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)

- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT ∗__s)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char ∗__s)
- template<class _Traits >
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char ∗__s)
- template<class _Traits >
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char ∗__s)
- template<class _Traits >
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char ∗__s)

5.368.1 Detailed Description

This is a Standard C++ Library header.

Definition in file ostream.

5.369 ostream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_OSTREAM_TCC**

Functions

- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char ∗__s)

5.369.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

Definition in file ostream.tcc.

5.370 ostream_insert.h File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT , typename _Traits >
  void **std::__ostream_fill** (basic_ostream< _CharT, _Traits > &__out, streamsize __n)
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::__ostream_insert** (basic_ostream< _CharT, _Traits > &__out, const _CharT ∗__s, streamsize __n)
- template<typename _CharT , typename _Traits >
  void **std::__ostream_write** (basic_ostream< _CharT, _Traits > &__out, const _CharT ∗__s, streamsize __n)

5.370.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

Definition in file ostream_insert.h.

5.371 ov_tree_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

  Ordered-vector tree associative-container.
- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >

  Conditional destructor.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CONST_NODE_ITERATOR_NAME**
- #define **PB_DS_OV_TREE_NAME**
- #define **PB_DS_OV_TREE_TRAITS_BASE**

5.371.1 Detailed Description

Contains an implementation class for ov_tree.

Definition in file ov_tree_map_.hpp.

5.372 pairing_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_P_HEAP_BASE**

5.372.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_.hpp.

5.373 par_loop.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template< typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result > _Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

5.373.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file par_loop.h.

5.374 parallel.h File Reference

5.374.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file parallel.h.

5.375 partial_sum.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator __gnu_parallel::__parallel_partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::value_type __value)
- template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::difference_type __n)

5.375.1 Detailed Description

Parallel implementation of std::partial_sum(), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file partial_sum.h.

5.376 partition.h File Reference

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_VOLATILE

Functions

- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter , typename _Predicate >
  std::iterator_traits< _RAIter >
  ::difference_type __gnu_parallel::__parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)

5.376.1 Detailed Description

Parallel implementation of std::partition(), std::nth_element(), and std::partial_sort(). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file partition.h.

5.376.2 Macro Definition Documentation

5.376.2.1 #define _GLIBCXX_VOLATILE

Decide whether to declare certain variables volatile.

Definition at line 43 of file partition.h.

Referenced by __gnu_parallel::__parallel_partition().

5.377 pat_trie_.hpp File Reference

Classes

- class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

  PATRICIA trie.
  This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_PAT_TRIE_NAME**
- #define **PB_DS_PAT_TRIE_TRAITS_BASE**
- #define **PB_DS_RECURSIVE_COUNT_LEAFS**(X)

5.377.1 Detailed Description

Contains an implementation class for a patricia tree.

Definition in file pat_trie_.hpp.

5.378 pat_trie_base.hpp File Reference

Classes

- struct __gnu_pbds::detail::pat_trie_base

  Base type for PATRICIA trees.

- class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

  Const iterator.
- struct __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >

  Head node for PATRICIA tree.
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >

  Internal node type, PATRICIA tree.
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator

  Constant child iterator.
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator

  Child iterator.
- class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

  Iterator.
- struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >

  Leaf node for PATRICIA tree.
- struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >

  Metadata base primary template.
- struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >

  Specialization for null metadata.
- struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >

  Node base.
- class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

  Node const iterator.
- class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

  Node iterator.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CONST_IT_C_DEC**
- #define **PB_DS_CONST_ODIR_IT_C_DEC**
- #define **PB_DS_IT_C_DEC**
- #define **PB_DS_ODIR_IT_C_DEC**
- #define **PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC**
- #define **PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC**

5.378.1 Detailed Description

Contains the base class for a patricia tree.

Definition in file pat_trie_base.hpp.

5.379 pod_char_traits.h File Reference

Classes

- struct __gnu_cxx::character< V, I, S >

  A POD class that serves as a character abstraction class.
- struct std::char_traits< __gnu_cxx::character< V, I, S > >

  char_traits<__gnu_cxx::character> specialization.

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename V , typename I , typename S >
  bool **__gnu_cxx::operator<** (const character< V, I, S > &lhs, const character< V, I, S > &rhs)
- template<typename V , typename I , typename S >
  bool **__gnu_cxx::operator==** (const character< V, I, S > &lhs, const character< V, I, S > &rhs)

5.379.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file pod_char_traits.h.

5.380 point_const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

  Const point-type iterator.

Namespaces

- namespace __gnu_pbds

5.380.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file binary_heap_/point_const_iterator.hpp.

5.381 point_const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >

  Const point-type iterator.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.381.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

Definition in file left_child_next_sibling_heap_/point_const_iterator.hpp.

5.382 point_const_iterator.hpp File Reference

Classes

- class point_const_iterator_

  Const point-type iterator.

5.382.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

Definition in file unordered_iterator/point_const_iterator.hpp.

5.383 point_iterator.hpp File Reference

Classes

- class point_iterator_

  Find type iterator.

5.383.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

Definition in file point_iterator.hpp.

5.384 point_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

  Const iterator.

- class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

  Iterator.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_TREE_CONST_IT_C_DEC**
- #define **PB_DS_TREE_CONST_ODIR_IT_C_DEC**
- #define **PB_DS_TREE_IT_C_DEC**
- #define **PB_DS_TREE_ODIR_IT_C_DEC**

5.384.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file point_iterators.hpp.

5.385 pointer.h File Reference

Classes

- struct __gnu_cxx::_Invalid_type
- class __gnu_cxx::_Pointer_adapter< _Storage_policy >
- class __gnu_cxx::_Relative_pointer_impl< _Tp >

  A storage policy for use with _Pointer_adapter<> which stores the pointer's address as an offset value which is relative to its own address.
- class __gnu_cxx::_Relative_pointer_impl< const _Tp >
- class __gnu_cxx::_Std_pointer_impl< _Tp >

  A storage policy for use with _Pointer_adapter<> which yields a standard pointer.
- struct __gnu_cxx::_Unqualified_type< _Tp >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_CXX_POINTER_ARITH_OPERATOR_SET**(INT_TYPE)
- #define **_GCC_CXX_POINTER_COMPARISON_OPERATION_SET**(OPERATOR)

Functions

- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator!=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator!=** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator**< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator**< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator**< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _CharT , typename _Traits , typename _StoreT >
  std::basic_ostream< _CharT,
  _Traits > & **__gnu_cxx::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator<=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator==** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator==** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp >
  bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator**> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator**> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)

- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1 , typename _Tp2 >
  bool **__gnu_cxx::operator>=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp >
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)

5.385.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable _Pointer_adapter for assisting in the development of custom pointer types that can be used with the standard containers via the allocator::pointer and allocator::const_pointer typedefs.

Definition in file pointer.h.

5.386 policy_access_fn_imps.hpp File Reference

5.386.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/policy_access_fn_imps.hpp.

5.387 policy_access_fn_imps.hpp File Reference

5.387.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/policy_access_fn_imps.hpp.

5.388 policy_access_fn_imps.hpp File Reference

5.388.1 Detailed Description

Contains implementations of cc_ht_map_'s policy access functions.

Definition in file cc_hash_table_map_/policy_access_fn_imps.hpp.

5.389 policy_access_fn_imps.hpp File Reference

5.389.1 Detailed Description

Contains implementations of gp_ht_map_'s policy agpess functions.

Definition in file gp_hash_table_map_/policy_access_fn_imps.hpp.

5.390 policy_access_fn_imps.hpp File Reference

5.390.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/policy_access_fn_imps.hpp.

5.391 policy_access_fn_imps.hpp File Reference

5.391.1 Detailed Description

Contains an implementation class for ov_tree.

Definition in file ov_tree_map_/policy_access_fn_imps.hpp.

5.392 policy_access_fn_imps.hpp File Reference

5.392.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/policy_access_fn_imps.hpp.

5.393 pool_allocator.h File Reference

Classes

- class __gnu_cxx::__pool_alloc< _Tp >

  Allocator using a memory pool with a single lock.
- class __gnu_cxx::__pool_alloc_base

  Base class for __pool_alloc.

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp >
  bool **__gnu_cxx::operator!=** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)
- template<typename _Tp >
  bool **__gnu_cxx::operator==** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)

5.393.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file pool_allocator.h.

5.394 postypes.h File Reference

Classes

- class std::fpos< _StateT >

  Class representing stream positions.

Namespaces

- namespace std

Typedefs

- typedef long long std::streamoff
- typedef fpos< mbstate_t > std::streampos
- typedef ptrdiff_t std::streamsize
- typedef fpos< mbstate_t > std::u16streampos
- typedef fpos< mbstate_t > std::u32streampos
- typedef fpos< mbstate_t > std::wstreampos

Functions

- template<typename _StateT >
  bool **std::operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _StateT >
  bool std::operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)

5.394.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

Definition in file postypes.h.

5.395 prefix_search_node_update_imp.hpp File Reference

5.395.1 Detailed Description

Contains an implementation of prefix_search_node_update.

Definition in file prefix_search_node_update_imp.hpp.

5.396 priority_queue.hpp File Reference

Classes

- class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

Namespaces

- namespace __gnu_pbds

5.396.1 Detailed Description

Contains priority_queues.

Definition in file priority_queue.hpp.

5.397 priority_queue_base_dispatch.hpp File Reference

Classes

- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >

  Specialization for binary_heap.
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >

  Specialization for binomial_heap.
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >

  Specialization for pairing_heap.
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >

  Specialization for rc_binary_heap.
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >

  Specialization for thin_heap.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_DEBUG_VERIFY**(_Cond)

5.397.1 Detailed Description

Contains an pqiative container dispatching base.

Definition in file priority_queue_base_dispatch.hpp.

5.398 probe_fn_base.hpp File Reference

Classes

- class __gnu_pbds::detail::probe_fn_base< _Alloc >

  Probe functor base.

Namespaces

- namespace __gnu_pbds

5.398.1 Detailed Description

Contains a probe policy base.

Definition in file probe_fn_base.hpp.

5.399 profiler.h File Reference

Classes

- struct __gnu_profile::__reentrance_guard

  Reentrance guard.

Namespaces

- namespace __gnu_profile

Macros

- #define **__profcxx_hashtable_construct**(__x...)
- #define **__profcxx_hashtable_construct2**(__x...)
- #define **__profcxx_hashtable_destruct**(__x...)
- #define **__profcxx_hashtable_destruct2**(__x...)
- #define **__profcxx_hashtable_resize**(__x...)
- #define **__profcxx_is_invalid**()
- #define **__profcxx_is_off**()
- #define **__profcxx_is_on**()
- #define **__profcxx_list_construct**(__x...)
- #define **__profcxx_list_construct2**(__x...)
- #define **__profcxx_list_destruct**(__x...)
- #define **__profcxx_list_destruct2**(__x...)
- #define **__profcxx_list_insert**(__x...)
- #define **__profcxx_list_invalid_operator**(__x...)
- #define **__profcxx_list_iterate**(__x...)
- #define **__profcxx_list_operation**(__x...)
- #define **__profcxx_list_rewind**(__x...)
- #define **__profcxx_map_to_unordered_map_construct**(__x...)
- #define **__profcxx_map_to_unordered_map_destruct**(__x...)

- #define **__profcxx_map_to_unordered_map_erase**(__x...)
- #define **__profcxx_map_to_unordered_map_find**(__x...)
- #define **__profcxx_map_to_unordered_map_insert**(__x...)
- #define **__profcxx_map_to_unordered_map_invalidate**(__x...)
- #define **__profcxx_map_to_unordered_map_iterate**(__x...)
- #define **__profcxx_report**()
- #define **__profcxx_turn_off**()
- #define **__profcxx_turn_on**()
- #define **__profcxx_vector_construct**(__x...)
- #define **__profcxx_vector_construct2**(__x...)
- #define **__profcxx_vector_destruct**(__x...)
- #define **__profcxx_vector_destruct2**(__x...)
- #define **__profcxx_vector_find**(__x...)
- #define **__profcxx_vector_insert**(__x...)
- #define **__profcxx_vector_invalid_operator**(__x...)
- #define **__profcxx_vector_iterate**(__x...)
- #define **__profcxx_vector_resize**(__x...)
- #define **__profcxx_vector_resize2**(__x...)
- #define **_GLIBCXX_PROFILE_DATA**(__name)
- #define **_GLIBCXX_PROFILE_DEFINE_DATA**(__type, __name, __initial_value...)
- #define **_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA**(__type, __name)
- #define **_GLIBCXX_PROFILE_MAX_STACK_DEPTH**
- #define **_GLIBCXX_PROFILE_MAX_STACK_DEPTH_ENV_VAR**
- #define **_GLIBCXX_PROFILE_MAX_WARN_COUNT**
- #define **_GLIBCXX_PROFILE_MAX_WARN_COUNT_ENV_VAR**
- #define **_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC**
- #define **_GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC_ENV_VAR**
- #define **_GLIBCXX_PROFILE_REENTRANCE_GUARD**(__x...)
- #define **_GLIBCXX_PROFILE_TRACE_ENV_VAR**
- #define **_GLIBCXX_PROFILE_TRACE_PATH_ROOT**

Functions

- bool **__gnu_profile::__is_invalid** ()
- bool **__gnu_profile::__is_off** ()
- bool **__gnu_profile::__is_on** ()
- void __gnu_profile::__report (void)
- void **__gnu_profile::__trace_hash_func_construct** (const void ∗)
- void **__gnu_profile::__trace_hash_func_destruct** (const void ∗, std::size_t, std::size_t, std::size_t)
- void **__gnu_profile::__trace_hashtable_size_construct** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_hashtable_size_destruct** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_hashtable_size_resize** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_list_to_set_construct** (const void ∗)
- void **__gnu_profile::__trace_list_to_set_destruct** (const void ∗)
- void **__gnu_profile::__trace_list_to_set_find** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_list_to_set_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_list_to_set_invalid_operator** (const void ∗)
- void **__gnu_profile::__trace_list_to_set_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_list_to_slist_construct** (const void ∗)
- void **__gnu_profile::__trace_list_to_slist_destruct** (const void ∗)

- void **__gnu_profile::__trace_list_to_slist_operation** (const void ∗)
- void **__gnu_profile::__trace_list_to_slist_rewind** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_construct** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_destruct** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_list_to_vector_invalid_operator** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_list_to_vector_resize** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_construct** (const void ∗)
- void **__gnu_profile::__trace_map_to_unordered_map_destruct** (const void ∗)
- void **__gnu_profile::__trace_map_to_unordered_map_erase** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_find** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_invalidate** (const void ∗)
- void **__gnu_profile::__trace_map_to_unordered_map_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_size_construct** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_size_destruct** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_size_resize** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_construct** (const void ∗)
- void **__gnu_profile::__trace_vector_to_list_destruct** (const void ∗)
- void **__gnu_profile::__trace_vector_to_list_find** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_invalid_operator** (const void ∗)
- void **__gnu_profile::__trace_vector_to_list_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_resize** (const void ∗, std::size_t, std::size_t)
- bool **__gnu_profile::__turn_off** ()
- bool **__gnu_profile::__turn_on** ()

5.399.1 Detailed Description

Interface of the profiling runtime library.

Definition in file profiler.h.

5.400 profiler_algos.h File Reference

Namespaces

- namespace __gnu_profile

Functions

- template<typename _InputIterator , typename _Function >
  _Function **__gnu_profile::__for_each** (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _Container >
  void **__gnu_profile::__insert_top_n** (_Container &__output, const typename _Container::value_type &__value, typename _Container::size_type __n)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator **__gnu_profile::__remove** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- template<typename _Container >
  void **__gnu_profile::__top_n** (const _Container &__input, _Container &__output, typename _Container::size_type __n)

5.400.1 Detailed Description

Algorithms used by the profile extension. This file is needed to avoid including <algorithm> or <bits/stl_algo.h>. Including those files would result in recursive includes. These implementations are oversimplified. In general, efficiency may be sacrificed to minimize maintenance overhead.

Definition in file profiler_algos.h.

5.401 profiler_container_size.h File Reference

Classes

- class __gnu_profile::__container_size_info
  A container size instrumentation line in the object table.
- class __gnu_profile::__container_size_stack_info
  A container size instrumentation line in the stack table.
- class __gnu_profile::__trace_container_size
  Container size instrumentation trace producer.

Namespaces

- namespace __gnu_profile

5.401.1 Detailed Description

Diagnostics for container sizes.

Definition in file profiler_container_size.h.

5.402 profiler_hash_func.h File Reference

Classes

- class __gnu_profile::__hashfunc_info
  A hash performance instrumentation line in the object table.
- class __gnu_profile::__hashfunc_stack_info
  A hash performance instrumentation line in the stack table.
- class __gnu_profile::__trace_hash_func
  Hash performance instrumentation producer.

Namespaces

- namespace __gnu_profile

Functions

- void **__gnu_profile::__trace_hash_func_construct** (const void ∗)
- void **__gnu_profile::__trace_hash_func_destruct** (const void ∗, std::size_t, std::size_t, std::size_t)
- void **__gnu_profile::__trace_hash_func_init** ()
- void **__gnu_profile::__trace_hash_func_report** (FILE ∗__f, __warning_vector_t &__warnings)

5.402.1 Detailed Description

Data structures to represent profiling traces.

Definition in file profiler_hash_func.h.

5.403 profiler_hashtable_size.h File Reference

Classes

- class __gnu_profile::__trace_hashtable_size

  Hashtable size instrumentation trace producer.

Namespaces

- namespace __gnu_profile

Functions

- void **__gnu_profile::__trace_hashtable_size_construct** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_hashtable_size_destruct** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_hashtable_size_init** ()
- void **__gnu_profile::__trace_hashtable_size_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_hashtable_size_resize** (const void ∗, std::size_t, std::size_t)

5.403.1 Detailed Description

Collection of hashtable size traces.

Definition in file profiler_hashtable_size.h.

5.404 profiler_list_to_slist.h File Reference

Namespaces

- namespace __gnu_profile

Functions

- void **__gnu_profile::__trace_list_to_slist_construct** (const void ∗)
- void **__gnu_profile::__trace_list_to_slist_destruct** (const void ∗)
- void **__gnu_profile::__trace_list_to_slist_init** ()
- void **__gnu_profile::__trace_list_to_slist_operation** (const void ∗)
- void **__gnu_profile::__trace_list_to_slist_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_list_to_slist_rewind** (const void ∗)

5.404.1 Detailed Description

Diagnostics for list to slist.

Definition in file profiler_list_to_slist.h.

5.405 profiler_list_to_vector.h File Reference

Classes

- class __gnu_profile::__list2vector_info

  A list-to-vector instrumentation line in the object table.

Namespaces

- namespace __gnu_profile

Functions

- void **__gnu_profile::__trace_list_to_vector_construct** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_destruct** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_init** ()
- void **__gnu_profile::__trace_list_to_vector_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_list_to_vector_invalid_operator** (const void ∗)
- void **__gnu_profile::__trace_list_to_vector_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_list_to_vector_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_list_to_vector_resize** (const void ∗, std::size_t, std::size_t)

5.405.1 Detailed Description

diagnostics for list to vector.

Definition in file profiler_list_to_vector.h.

5.406 profiler_map_to_unordered_map.h File Reference

Classes

- class __gnu_profile::__map2umap_info

  A map-to-unordered_map instrumentation line in the object table.

- class __gnu_profile::__map2umap_stack_info

  A map-to-unordered_map instrumentation line in the stack table.
- class __gnu_profile::__trace_map2umap

  Map-to-unordered_map instrumentation producer.

Namespaces

- namespace __gnu_profile

Functions

- int **__gnu_profile::__log2** (std::size_t __size)
- float **__gnu_profile::__map_erase_cost** (std::size_t __size)
- float **__gnu_profile::__map_find_cost** (std::size_t __size)
- float **__gnu_profile::__map_insert_cost** (std::size_t __size)
- void **__gnu_profile::__trace_map_to_unordered_map_construct** (const void ∗)
- void **__gnu_profile::__trace_map_to_unordered_map_destruct** (const void ∗)
- void **__gnu_profile::__trace_map_to_unordered_map_erase** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_find** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_init** ()
- void **__gnu_profile::__trace_map_to_unordered_map_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_invalidate** (const void ∗)
- void **__gnu_profile::__trace_map_to_unordered_map_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_map_to_unordered_map_report** (FILE ∗__f, __warning_vector_t &__warnings)

5.406.1 Detailed Description

Diagnostics for map to unordered_map.

Definition in file profiler_map_to_unordered_map.h.

5.407 profiler_node.h File Reference

Classes

- class __gnu_profile::__object_info_base

  Base class for a line in the object table.
- class __gnu_profile::__stack_hash

  Hash function for summary trace using call stack as index.
- class __gnu_profile::__stack_info_base< __object_info >

  Base class for a line in the stack table.

Namespaces

- namespace __gnu_profile

Typedefs

- typedef void ∗ **__gnu_profile::__instruction_address_t**
- typedef const void ∗ **__gnu_profile::__object_t**
- typedef std::::vector
  < __instruction_address_t > **__gnu_profile::__stack_npt**
- typedef __stack_npt ∗ **__gnu_profile::__stack_t**

Functions

- __stack_t **__gnu_profile::__get_stack** ()
- std::size_t **__gnu_profile::__size** (__stack_t __stack)
- std::size_t **__gnu_profile::__stack_max_depth** ()
- void **__gnu_profile::__write** (FILE ∗__f, __stack_t __stack)

5.407.1 Detailed Description

Data structures to represent a single profiling event.

Definition in file profiler_node.h.

5.408 profiler_state.h File Reference

Namespaces

- namespace __gnu_profile

Enumerations

- enum **__state_type** { **__ON**, **__OFF**, **__INVALID** }

Functions

- bool **__gnu_profile::__is_invalid** ()
- bool **__gnu_profile::__is_off** ()
- bool **__gnu_profile::__is_on** ()
- bool **__gnu_profile::__turn** (__state_type __s)
- bool **__gnu_profile::__turn_off** ()
- bool **__gnu_profile::__turn_on** ()
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__state_type, __state, __INVALID)

5.408.1 Detailed Description

Global profiler state.

Definition in file profiler_state.h.

5.409 profiler_trace.h File Reference

Classes

- class __gnu_profile::__trace_base< __object_info, __stack_info >

  Base class for all trace producers.
- struct __gnu_profile::__warning_data

  Representation of a warning.

Namespaces

- namespace __gnu_profile

Macros

- #define **_GLIBCXX_IMPL_UNORDERED_MAP**

Typedefs

- typedef std::::vector < __cost_factor ∗ > **__gnu_profile::__cost_factor_vector**
- typedef std::::unordered_map < std::string, std::string > __gnu_profile::__env_t
- typedef std::::vector < __warning_data > **__gnu_profile::__warning_vector_t**

Functions

- std::size_t **__gnu_profile::__env_to_size_t** (const char ∗__env_var, std::size_t __default_value)
- int **__gnu_profile::__log_magnitude** (float __f)
- std::size_t **__gnu_profile::__max_mem** ()
- FILE ∗ **__gnu_profile::__open_output_file** (const char ∗__extension)
- bool __gnu_profile::__profcxx_init ()
- void **__gnu_profile::__profcxx_init_unconditional** ()
- void **__gnu_profile::__read_cost_factors** ()
- void __gnu_profile::__report (void)
- void **__gnu_profile::__set_cost_factors** ()
- void **__gnu_profile::__set_max_mem** ()
- void **__gnu_profile::__set_max_stack_trace_depth** ()
- void **__gnu_profile::__set_max_warn_count** ()
- void **__gnu_profile::__set_trace_path** ()
- std::size_t **__gnu_profile::__stack_max_depth** ()
- void **__gnu_profile::__trace_hash_func_init** ()
- void **__gnu_profile::__trace_hash_func_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_hashtable_size_init** ()
- void **__gnu_profile::__trace_hashtable_size_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_list_to_slist_init** ()
- void **__gnu_profile::__trace_list_to_slist_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_list_to_vector_init** ()

- void **__gnu_profile::__trace_list_to_vector_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_map_to_unordered_map_init** ()
- void **__gnu_profile::__trace_map_to_unordered_map_report** (FILE ∗__f, __warning_vector_t &__warnings)
- void **__gnu_profile::__trace_vector_size_init** ()
- void **__gnu_profile::__trace_vector_size_report** (FILE ∗, __warning_vector_t &)
- void **__gnu_profile::__trace_vector_to_list_init** ()
- void **__gnu_profile::__trace_vector_to_list_report** (FILE ∗, __warning_vector_t &)
- void **__gnu_profile::__write_cost_factors** ()
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_hash_func ∗, _S_hash_func, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_hashtable_size ∗, _S_hashtable_size, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_map2umap ∗, _S_map2umap, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_vector_size ∗, _S_vector_size, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_vector_to_list ∗, _S_vector_to_list, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_list_to_slist ∗, _S_list_to_slist, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__trace_list_to_vector ∗, _S_list_to_vector, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_shift_cost_factor,{"__vector_shift_cost_factor", 1.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_iterate_cost_factor,{"__vector_iterate_cost_factor", 1.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __vector_resize_cost_factor,{"__vector_resize_cost_factor", 1.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __list_shift_cost_factor,{"__list_shift_cost_factor", 0.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __list_iterate_cost_factor,{"__list_iterate_cost_factor", 10.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __list_resize_cost_factor,{"__list_resize_cost_factor", 0.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_insert_cost_factor,{"__map_insert_cost_factor", 1.5})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_erase_cost_factor,{"__map_erase_cost_factor", 1.5})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_find_cost_factor,{"__map_find_cost_factor", 1})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __map_iterate_cost_factor,{"__map_iterate_cost_factor", 2.3})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_insert_cost_factor,{"__umap_insert_cost_factor", 12.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_erase_cost_factor,{"__umap_erase_cost_factor", 12.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_find_cost_factor,{"__umap_find_cost_factor", 10.0})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor, __umap_iterate_cost_factor,{"__umap_iterate_cost_factor", 1.7})
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (__cost_factor_vector ∗, __cost_factors, 0)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (const char ∗, _S_trace_file_name, _GLIBCXX_PROFILE_TRACE_PATH_ROOT)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (std::size_t, _S_max_warn_count, _GLIBCXX_PROFILE_MAX_WARN_COUNT)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (std::size_t, _S_max_stack_depth, _GLIBCXX_PROFILE_MAX_STACK_DEPTH)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_DATA** (std::size_t, _S_max_mem, _GLIBCXX_PROFILE_MEM_PER_DIAGNOSTIC)
- **__gnu_profile::_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA** (__env_t, __env)
- __gnu_profile::_GLIBCXX_PROFILE_DEFINE_UNINIT_DATA (__gnu_cxx::__mutex, __global_lock)

5.409.1 Detailed Description

Data structures to represent profiling traces.

Definition in file profiler_trace.h.

5.410 profiler_vector_size.h File Reference

Classes

- class __gnu_profile::__trace_vector_size

  Hashtable size instrumentation trace producer.

Namespaces

- namespace __gnu_profile

Functions

- void **__gnu_profile::__trace_vector_size_construct** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_size_destruct** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_size_init** ()
- void **__gnu_profile::__trace_vector_size_report** (FILE ∗, __warning_vector_t &)
- void **__gnu_profile::__trace_vector_size_resize** (const void ∗, std::size_t, std::size_t)

5.410.1 Detailed Description

Collection of vector size traces.

Definition in file profiler_vector_size.h.

5.411 profiler_vector_to_list.h File Reference

Classes

- class __gnu_profile::__trace_vector_to_list

  Vector-to-list instrumentation producer.
- class __gnu_profile::__vector2list_info

  A vector-to-list instrumentation line in the object table.
- class __gnu_profile::__vector2list_stack_info

  A vector-to-list instrumentation line in the stack table.

Namespaces

- namespace __gnu_profile

Functions

- void **__gnu_profile::__trace_vector_to_list_construct** (const void ∗)
- void **__gnu_profile::__trace_vector_to_list_destruct** (const void ∗)
- void **__gnu_profile::__trace_vector_to_list_find** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_init** ()
- void **__gnu_profile::__trace_vector_to_list_insert** (const void ∗, std::size_t, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_invalid_operator** (const void ∗)
- void **__gnu_profile::__trace_vector_to_list_iterate** (const void ∗, std::size_t)
- void **__gnu_profile::__trace_vector_to_list_report** (FILE ∗, __warning_vector_t &)
- void **__gnu_profile::__trace_vector_to_list_resize** (const void ∗, std::size_t, std::size_t)

5.411.1 Detailed Description

diagnostics for vector to list.

Definition in file profiler_vector_to_list.h.

5.412 ptr_traits.h File Reference

Classes

- struct std::pointer_traits< _Ptr >

  Uniform interface to all pointer-like types.
- struct std::pointer_traits< _Tp ∗ >

  Partial specialization for built-in pointers.

Namespaces

- namespace std

5.412.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file ptr_traits.h.

5.413 quadratic_probe_fn_imp.hpp File Reference

5.413.1 Detailed Description

Contains a probe policy implementation

Definition in file quadratic_probe_fn_imp.hpp.

5.414 queue File Reference

Macros

- #define **_GLIBCXX_QUEUE**

5.414.1 Detailed Description

This is a Standard C++ Library header.

Definition in file queue.

5.415 queue.h File Reference

Classes

- class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >

  Double-ended queue of bounded size, allowing lock-free atomic access. push_front() and pop_front() must not be called concurrently to each other, while pop_back() can be called concurrently at all times. `empty()`, `size()`, *and* `top()` *are intentionally not provided. Calling them would not make sense in a concurrent setting.*

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_VOLATILE

5.415.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file queue.h.

5.415.2 Macro Definition Documentation

5.415.2.1 #define _GLIBCXX_VOLATILE

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file queue.h.

5.416 quicksort.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)

- template<typename _RAIter , typename _Compare >
  std::iterator_traits< _RAIter >
  ::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)

5.416.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file quicksort.h.

5.417 r_erase_fn_imps.hpp File Reference

5.417.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/r_erase_fn_imps.hpp.

5.418 r_erase_fn_imps.hpp File Reference

5.418.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file pat_trie_/r_erase_fn_imps.hpp.

5.419 random File Reference

Macros

- #define **_GLIBCXX_RANDOM**

5.419.1 Detailed Description

This is a Standard C++ Library header.

Definition in file random.

5.420 random.h File Reference

Classes

- class std::bernoulli_distribution

  A Bernoulli random number distribution.
- struct std::bernoulli_distribution::param_type
- class std::binomial_distribution< _IntType >

  A discrete binomial random number distribution.

- struct std::binomial_distribution< _IntType >::param_type
- class std::cauchy_distribution< _RealType >

  A cauchy_distribution random number distribution.
- struct std::cauchy_distribution< _RealType >::param_type
- class std::chi_squared_distribution< _RealType >

  A chi_squared_distribution random number distribution.
- struct std::chi_squared_distribution< _RealType >::param_type
- class std::discard_block_engine< _RandomNumberEngine, __p, __r >
- class std::discrete_distribution< _IntType >

  A discrete_distribution random number distribution.
- struct std::discrete_distribution< _IntType >::param_type
- class std::exponential_distribution< _RealType >

  An exponential continuous distribution for random numbers.
- struct std::exponential_distribution< _RealType >::param_type
- class std::extreme_value_distribution< _RealType >

  A extreme_value_distribution random number distribution.
- struct std::extreme_value_distribution< _RealType >::param_type
- class std::fisher_f_distribution< _RealType >

  A fisher_f_distribution random number distribution.
- struct std::fisher_f_distribution< _RealType >::param_type
- class std::gamma_distribution< _RealType >

  A gamma continuous distribution for random numbers.
- struct std::gamma_distribution< _RealType >::param_type
- class std::geometric_distribution< _IntType >

  A discrete geometric random number distribution.
- struct std::geometric_distribution< _IntType >::param_type
- class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
- class std::linear_congruential_engine< _UIntType, __a, __c, __m >

  A model of a linear congruential random number generator.
- class std::lognormal_distribution< _RealType >

  A lognormal_distribution random number distribution.
- struct std::lognormal_distribution< _RealType >::param_type
- class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
- class std::negative_binomial_distribution< _IntType >

  A negative_binomial_distribution random number distribution.
- struct std::negative_binomial_distribution< _IntType >::param_type
- class std::normal_distribution< _RealType >

  A normal continuous distribution for random numbers.
- struct std::normal_distribution< _RealType >::param_type
- class std::piecewise_constant_distribution< _RealType >

  A piecewise_constant_distribution random number distribution.
- struct std::piecewise_constant_distribution< _RealType >::param_type
- class std::piecewise_linear_distribution< _RealType >

  A piecewise_linear_distribution random number distribution.
- struct std::piecewise_linear_distribution< _RealType >::param_type
- class std::poisson_distribution< _IntType >

  A discrete Poisson random number distribution.

- struct std::poisson_distribution< _IntType >::param_type
- class std::random_device
- class std::seed_seq

  The seed_seq class generates sequences of seeds for random number generators.
- class std::shuffle_order_engine< _RandomNumberEngine, __k >

  Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits __w.
- class std::student_t_distribution< _RealType >

  A student_t_distribution random number distribution.
- struct std::student_t_distribution< _RealType >::param_type
- class std::uniform_int_distribution< _IntType >

  Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.
- struct std::uniform_int_distribution< _IntType >::param_type
- class std::uniform_real_distribution< _RealType >

  Uniform continuous distribution for random numbers.
- struct std::uniform_real_distribution< _RealType >::param_type
- class std::weibull_distribution< _RealType >

  A weibull_distribution random number distribution.
- struct std::weibull_distribution< _RealType >::param_type

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine
  < minstd_rand0, 256 > **std::knuth_b**
- typedef
  linear_congruential_engine
  < uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand
- typedef
  linear_congruential_engine
  < uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0
- typedef
  mersenne_twister_engine
  < uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937
- typedef
  mersenne_twister_engine
  < uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64
- typedef discard_block_engine
  < ranlux24_base, 223, 23 > **std::ranlux24**
- typedef
  subtract_with_carry_engine
  < uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**

- typedef discard_block_engine
  < ranlux48_base, 389, 11 > **std::ranlux48**
- typedef
  subtract_with_carry_engine
  < uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator >
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m>
  bool std::operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > &__rhs)
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
  bool std::operator!= (const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)
- template<typename _UIntType , size_t __w, size_t __s, size_t __r>
  bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)
- template<typename _RandomNumberEngine , size_t __p, size_t __r>
  bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)
- template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
  bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)
- template<typename _RandomNumberEngine , size_t __k>
  bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)
- template<typename _IntType >
  bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _RealType > &__d2)

- template<typename _RealType >
  bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)
- bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)
- template<typename _IntType >
  bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)
- template<typename _RealType >
  bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)
- template<typename _RandomNumberEngine , size_t __w, typename _UIntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)

- template<typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)

5.420.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

Definition in file random.h.

5.421 random.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_RANDOM_TCC**

Functions

- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator **std::__detail::__normalize** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__factor)
- template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator >
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _UIntType , size_t __w, size_t __s, size_t __r, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template<typename _RandomNumberEngine , size_t __p, size_t __r, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _RandomNumberEngine , size_t __k, typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)

- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & **std::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)

- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)
- template<typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_ostream< _CharT,
  _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _RealType >
  bool **std::operator==** (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _UIntType , size_t __w, size_t __s, size_t __r, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template<typename _RandomNumberEngine , size_t __p, size_t __r, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _RandomNumberEngine , size_t __k, typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType > &__x)

- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)

- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT,
_Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)

5.421.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

Definition in file bits/random.tcc.

5.422 random.tcc File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_RANDOM_TCC**

Functions

- template<typename _UIntType , size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT,
_Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)

- template<size_t _Dimen, typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)
- template<typename _UIntType , size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>
bool **__gnu_cxx::operator==** (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)
- template<size_t _Dimen, typename _RealType >
bool **__gnu_cxx::operator==** (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)
- template<typename _UIntType , size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)
- template<size_t _Dimen, typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)

- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)
- template<typename _RealType , typename _CharT , typename _Traits >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)

5.422.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/random>.

Definition in file ext/random.tcc.

5.423 random_number.h File Reference

Classes

- class __gnu_parallel::_RandomNumber

  Random number generator, based on the Mersenne twister.

Namespaces

- namespace __gnu_parallel

5.423.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file random_number.h.

5.424 random_shuffle.h File Reference

Classes

- struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >

  Data known to every thread participating in __gnu_parallel::__parallel_random_shuffle().
- struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

  Local data for a thread participating in __gnu_parallel::__parallel_random_shuffle().

Namespaces

- namespace __gnu_parallel

Typedefs

- typedef unsigned short __gnu_parallel::_BinIndex

Functions

- template<typename _RAIter , typename _RandomNumberGenerator >
  void __gnu_parallel::__parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __gnu_parallel::__parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)
- template<typename _RandomNumberGenerator >
  int __gnu_parallel::__random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Tp >
  _Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)
- template<typename _RAIter , typename _RandomNumberGenerator >
  void __gnu_parallel::__sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)

5.424.1 Detailed Description

Parallel implementation of std::random_shuffle(). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file random_shuffle.h.

5.425 range_access.h File Reference

Namespaces

- namespace std

Functions

- template<class _Container >
  auto std::begin (_Container &__cont) -> decltype(__cont.begin())
- template<class _Container >
  auto std::begin (const _Container &__cont) -> decltype(__cont.begin())
- template<class _Tp , size_t _Nm>
  _Tp ∗ std::begin (_Tp(&__arr)[_Nm])
- template<class _Container >
  auto std::end (_Container &__cont) -> decltype(__cont.end())
- template<class _Container >
  auto std::end (const _Container &__cont) -> decltype(__cont.end())
- template<class _Tp , size_t _Nm>
  _Tp ∗ std::end (_Tp(&__arr)[_Nm])

5.425.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

Definition in file range_access.h.

5.426 ranged_hash_fn.hpp File Reference

Classes

- class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
- class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
- class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
- class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
- class ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >

  Primary template.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.426.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

Definition in file ranged_hash_fn.hpp.

5.427 ranged_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
- class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
- class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
- class ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >

  Primary template.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.427.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probeing.

Definition in file ranged_probe_fn.hpp.

5.428 ratio File Reference

Classes

- struct std::ratio< _Num, _Den >

  Provides compile-time rational arithmetic.
- struct std::ratio_equal< _R1, _R2 >

  ratio_equal
- struct std::ratio_not_equal< _R1, _R2 >

  ratio_not_equal

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_RATIO**

Typedefs

- template< typename _R1 , typename _R2 >
  using std::ratio_divide = typename __ratio_divide< _R1, _R2 >::type
- template< typename _R1 , typename _R2 >
  using std::ratio_multiply = typename __ratio_multiply< _R1, _R2 >::type

5.428.1 Detailed Description

This is a Standard C++ Library header.

Definition in file ratio.

5.429 ratio File Reference

Namespaces

- namespace std
- namespace std::tr2

5.429.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file tr2/ratio.

5.430 rb_tree File Reference

Classes

- struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_RB_TREE**

5.430.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file rb_tree.

5.431 rb_tree_.hpp File Reference

Classes

- class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

  Red-Black tree.
  This implementation uses an idea from the SGI STL (using a header node which is needed for efficient iteration).

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_RB_TREE_BASE**
- #define **PB_DS_RB_TREE_BASE_NAME**
- #define **PB_DS_RB_TREE_NAME**
- #define **PB_DS_STRUCT_ONLY_ASSERT_VALID**(X)

5.431.1 Detailed Description

Contains an implementation for Red Black trees.

Definition in file rb_tree_.hpp.

5.432 rc.hpp File Reference

Classes

- class __gnu_pbds::detail::rc< _Node, _Alloc >

  Redundant binary counter.

Namespaces

- namespace __gnu_pbds

5.432.1 Detailed Description

Contains a redundant (binary counter).

Definition in file rc.hpp.

5.433 rc_binomial_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_RC_C_DEC**

5.433.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

Definition in file rc_binomial_heap_.hpp.

5.434 rc_string_base.h File Reference

Classes

- class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >

Namespaces

- namespace __gnu_cxx

5.434.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

Definition in file rc_string_base.h.

5.435 regex File Reference

Macros

- #define **_GLIBCXX_REGEX**

5.435.1 Detailed Description

This is a Standard C++ Library header.

Definition in file regex.

5.436 regex.h File Reference

Classes

- class std::basic_regex< _Ch_type, _Rx_traits >

- class std::match_results< _Bi_iter, _Alloc >

  The results of a match or search operation.
- class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- struct std::regex_traits< _Ch_type >

  Class regex_traits. Describes aspects of a regular expression.
- class std::sub_match< _BiIter >

Namespaces

- namespace std

Typedefs

- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  using **std::__sub_match_string** = basic_string< typename iterator_traits< _Bi_iter >::value_type, _Ch_traits, _Ch_alloc >
- typedef match_results< const char ∗ > **std::cmatch**
- typedef regex_iterator< const char ∗ > **std::cregex_iterator**
- typedef regex_token_iterator < const char ∗ > std::cregex_token_iterator
- typedef sub_match< const char ∗ > std::csub_match
- typedef basic_regex< char > std::regex
- typedef match_results < string::const_iterator > **std::smatch**
- typedef regex_iterator < string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator < string::const_iterator > std::sregex_token_iterator
- typedef sub_match < string::const_iterator > std::ssub_match
- typedef match_results< const wchar_t ∗ > **std::wcmatch**
- typedef regex_iterator< const wchar_t ∗ > **std::wcregex_iterator**
- typedef regex_token_iterator < const wchar_t ∗ > std::wcregex_token_iterator
- typedef sub_match< const wchar_t ∗ > std::wcsub_match
- typedef basic_regex< wchar_t > std::wregex
- typedef match_results < wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator < wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator < wstring::const_iterator > std::wsregex_token_iterator
- typedef sub_match < wstring::const_iterator > std::wssub_match

Functions

- template<typename _Bi_iter >
  const sub_match< _Bi_iter > & **std::__unmatched_sub** ()
- template<typename _BiIter >
  bool std::operator!= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Bi_iter , class _Alloc >
  bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _BiIter >
  bool std::operator< (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator< (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter >
  basic_ostream< _Ch_type,
  _Ch_traits > & std::operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)
- template<typename _BiIter >
  bool std::operator<= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)

- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator<= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _BiIter >
  bool std::operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const ∗__rhs)
- template<typename _Bi_iter >
  bool std::operator== (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Bi_iter , typename _Alloc >
  bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _BiIter >
  bool std::operator> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const ∗__lhs, const sub_match< _Bi_iter > &__rhs)

- template<typename _Bi_iter >
  bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)
- template<typename _Bi_iter >
  bool std::operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _BiIter >
  bool std::operator>= (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
  bool std::operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter , class _Ch_traits , class _Ch_alloc >
  bool std::operator>= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const *__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)
- template<typename _Bi_iter >
  bool std::operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Ch_type , typename _Rx_traits >
  void std::swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)
- template<typename _Bi_iter , typename _Alloc >
  void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs)

Matching, Searching, and Replacing

- template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , typename _Alloc , typename _Rx_traits >
  bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , class _Rx_traits >
  bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)

- template<typename _Ch_traits , typename _Str_allocator , typename _Ch_type , typename _Rx_traits >
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type , class _Alloc , class _Rx_traits >
  bool std::regex_search (const _Ch_type ∗__s, match_results< const _Ch_type ∗, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (const _Ch_type ∗__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename _Rx_traits >
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type >
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits , typename _Ch_type >
  basic_string< _Ch_type > std::regex_replace (const basic_string< _Ch_type > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

5.436.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex.h.

5.437 regex_compiler.h File Reference

Classes

- class std::__detail::_Compiler< _InIter, _TraitsT >

  Builds an NFA from an input iterator interval.
- class std::__detail::_Scanner< _InputIterator >

  struct _Scanner. Scans an input range for regex tokens.
- struct std::__detail::_Scanner_base

  Base class for scanner.

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _InIter , typename _TraitsT >
  _AutomatonPtr **std::__detail::__compile** (const _InIter &__b, const _InIter &__e, _TraitsT &__t, regex_constants::syntax_option_type __f)

5.437.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_compiler.h.

5.438 regex_constants.h File Reference

Namespaces

- namespace std
- namespace std::regex_constants

5.1 Regular Expression Syntax Options

- enum std::regex_constants::__syntax_option {
  _S_icase, **_S_nosubs**, **_S_optimize**, **_S_collate**,
  _S_ECMAScript, **_S_basic**, **_S_extended**, **_S_awk**,
  _S_grep, **_S_egrep**, **_S_syntax_last** }
- typedef unsigned int std::regex_constants::syntax_option_type
- constexpr syntax_option_type std::regex_constants::icase
- constexpr syntax_option_type std::regex_constants::nosubs
- constexpr syntax_option_type std::regex_constants::optimize
- constexpr syntax_option_type std::regex_constants::collate
- constexpr syntax_option_type std::regex_constants::ECMAScript
- constexpr syntax_option_type std::regex_constants::basic
- constexpr syntax_option_type std::regex_constants::extended
- constexpr syntax_option_type std::regex_constants::awk
- constexpr syntax_option_type std::regex_constants::grep
- constexpr syntax_option_type std::regex_constants::egrep

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum std::regex_constants::__match_flag {
  _S_not_bol, **_S_not_eol**, **_S_not_bow**, **_S_not_eow**,
  _S_any, **_S_not_null**, **_S_continuous**, **_S_prev_avail**,
  _S_sed, **_S_no_copy**, **_S_first_only**, **_S_match_flag_last** }
- typedef std::bitset
  < _S_match_flag_last > std::regex_constants::match_flag_type
- constexpr match_flag_type std::regex_constants::match_default
- constexpr match_flag_type std::regex_constants::match_not_bol
- constexpr match_flag_type std::regex_constants::match_not_eol
- constexpr match_flag_type std::regex_constants::match_not_bow
- constexpr match_flag_type std::regex_constants::match_not_eow
- constexpr match_flag_type std::regex_constants::match_any
- constexpr match_flag_type std::regex_constants::match_not_null
- constexpr match_flag_type std::regex_constants::match_continuous
- constexpr match_flag_type std::regex_constants::match_prev_avail
- constexpr match_flag_type std::regex_constants::format_default
- constexpr match_flag_type std::regex_constants::format_sed
- constexpr match_flag_type std::regex_constants::format_no_copy
- constexpr match_flag_type std::regex_constants::format_first_only

5.438.1 Detailed Description

Constant definitions for the std regex library. This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_constants.h.

5.439 regex_cursor.h File Reference

Classes

- struct std::__detail::_PatternCursor

  ABC for pattern matching.
- class std::__detail::_SpecializedCursor< _FwdIterT >

  Provides a cursor into the specific target string.

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _FwdIterT >
  _SpecializedCursor< _FwdIterT > **std::__detail::__cursor** (const _FwdIterT &__b, const _FwdIterT __e)

5.439.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_cursor.h.

5.440 regex_error.h File Reference

Classes

- class std::regex_error

  A regular expression exception class.
  The regular expression library throws objects of this class on error.

Namespaces

- namespace std
- namespace std::regex_constants

Functions

- void **std::__throw_regex_error** (regex_constants::error_type __ecode)

5.3 Error Types

- enum std::regex_constants::error_type {
  _S_error_collate, **_S_error_ctype**, **_S_error_escape**, **_S_error_backref**,
  _S_error_brack, **_S_error_paren**, **_S_error_brace**, **_S_error_badbrace**,
  _S_error_range, **_S_error_space**, **_S_error_badrepeat**, **_S_error_complexity**,
  _S_error_stack, **_S_error_last** }
- constexpr error_type std::regex_constants::error_collate (_S_error_collate)
- constexpr error_type std::regex_constants::error_ctype (_S_error_ctype)
- constexpr error_type std::regex_constants::error_escape (_S_error_escape)
- constexpr error_type std::regex_constants::error_backref (_S_error_backref)
- constexpr error_type std::regex_constants::error_brack (_S_error_brack)
- constexpr error_type std::regex_constants::error_paren (_S_error_paren)
- constexpr error_type std::regex_constants::error_brace (_S_error_brace)
- constexpr error_type std::regex_constants::error_badbrace (_S_error_badbrace)
- constexpr error_type std::regex_constants::error_range (_S_error_range)
- constexpr error_type std::regex_constants::error_space (_S_error_space)
- constexpr error_type std::regex_constants::error_badrepeat (_S_error_badrepeat)
- constexpr error_type std::regex_constants::error_complexity (_S_error_complexity)
- constexpr error_type std::regex_constants::error_stack (_S_error_stack)

5.440.1 Detailed Description

Error and exception objects for the std regex library. This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_error.h.

5.441 regex_grep_matcher.h File Reference

Classes

- class std::__detail::_Grep_matcher

  Executes a regular expression NFA/DFA over a range using a variant of the parallel execution algorithm featured in the grep utility, modified to use Laurikari tags.
- class std::__detail::_SpecializedResults< _FwdIterT, _Alloc >

  A _Results facade specialized for wrapping a templated match_results.
- class std::match_results< _Bi_iter, _Alloc >

  The results of a match or search operation.
- class std::sub_match< _BiIter >

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- typedef std::stack< _StateIdT,
  std::vector< _StateIdT > > std::__detail::_StateStack

5.441.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_grep_matcher.h.

5.442 regex_grep_matcher.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

5.442.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_grep_matcher.tcc.

5.443 regex_nfa.h File Reference

Classes

- class std::__detail::_Automaton

Base class for, um, automata. Could be an NFA or a DFA. Your choice.

- struct std::__detail::_CharMatcher< _InIterT, _TraitsT >

  Matches a single character.
- struct std::__detail::_EndTagger< _FwdIterT, _TraitsT >

  End state tag.
- class std::__detail::_Nfa

  struct _Nfa
- struct std::__detail::_RangeMatcher< _InIterT, _TraitsT >

  Matches a character range (bracket expression)
- struct std::__detail::_Results

  Provides a generic facade for a templated match_results.
- struct std::__detail::_StartTagger< _FwdIterT, _TraitsT >

  Start state tag.
- struct std::__detail::_State

  struct _State
- class std::__detail::_StateSeq

  Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- typedef std::shared_ptr < _Automaton > std::__detail::_AutomatonPtr
- typedef std::function< bool(const _PatternCursor &)> std::__detail::_Matcher
- typedef int std::__detail::_StateIdT
- typedef std::set< _StateIdT > std::__detail::_StateSet
- typedef std::function< void(const _PatternCursor &, _Results &)> std::__detail::_Tagger

Enumerations

- enum std::__detail::_Opcode { **_S_opcode_unknown**, **_S_opcode_alternative**, **_S_opcode_subexpr_begin**, **_S_opcode_subexpr_end**, **_S_opcode_match**, **_S_opcode_accept** }

Functions

- bool std::__detail::_AnyMatcher (const _PatternCursor &)

Variables

- static const _StateIdT std::__detail::_S_invalid_state_id

5.443.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_nfa.h.

5.444 regex_nfa.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

5.444.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

Definition in file regex_nfa.tcc.

5.445 resize_fn_imps.hpp File Reference

5.445.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions.

Definition in file cc_hash_table_map_/resize_fn_imps.hpp.

5.446 resize_fn_imps.hpp File Reference

5.446.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions.

Definition in file gp_hash_table_map_/resize_fn_imps.hpp.

5.447 resize_no_store_hash_fn_imps.hpp File Reference

5.447.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions, when the hash value is not stored.

Definition in file cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp.

5.448 resize_no_store_hash_fn_imps.hpp File Reference

5.448.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions, when the hash value is not stored.

Definition in file gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp.

5.449 resize_policy.hpp File Reference

Classes

- class __gnu_pbds::detail::resize_policy< _Tp >

  Resize policy for binary heap.

Namespaces

- namespace __gnu_pbds

5.449.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file resize_policy.hpp.

5.450 resize_store_hash_fn_imps.hpp File Reference

5.450.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions, when the hash value is stored.

Definition in file cc_hash_table_map_/resize_store_hash_fn_imps.hpp.

5.451 resize_store_hash_fn_imps.hpp File Reference

5.451.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions, when the hash value is stored.

Definition in file gp_hash_table_map_/resize_store_hash_fn_imps.hpp.

5.452 rope File Reference

Classes

- class __gnu_cxx::rope< _CharT, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::__detail
- namespace std
- namespace std::tr1

Macros

- #define **__GC_CONST**
- #define **__ROPE_DEFINE_ALLOC**(_Tp, __name)
- #define **__ROPE_DEFINE_ALLOC**(_Tp, __name)
- #define **__ROPE_DEFINE_ALLOCS**(__a)
- #define **__STATIC_IF_SGI_ALLOC**
- #define **__STL_FREE_STRING**(__s, __l, __a)
- #define **__STL_ROPE_FROM_UNOWNED_CHAR_PTR**(__s, __size, __a)
- #define **_ROPE**

Typedefs

- typedef rope< char > **__gnu_cxx::crope**
- typedef rope< wchar_t > **__gnu_cxx::wrope**

Enumerations

- enum { **_S_max_rope_depth** }
- enum **_Tag** { **_S_leaf**, **_S_concat**, **_S_substringfn**, **_S_function** }

Functions

- crope::reference **__gnu_cxx::__mutable_reference_at** (crope &__c, size_t __i)
- template<typename _ForwardIterator , typename _Allocator >
  void **__gnu_cxx::_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)
- template<typename _ForwardIterator , typename _Tp >
  void **__gnu_cxx::_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp >)
- template<class _CharT >
  void **__gnu_cxx::_S_cond_store_eos** (_CharT &)
- void **__gnu_cxx::_S_cond_store_eos** (char &__c)
- void **__gnu_cxx::_S_cond_store_eos** (wchar_t &__c)
- template<class _CharT >
  _CharT **__gnu_cxx::_S_eos** (_CharT ∗)
- template<class _CharT >
  bool **__gnu_cxx::_S_is_basic_char_type** (_CharT ∗)
- bool **__gnu_cxx::_S_is_basic_char_type** (char ∗)
- bool **__gnu_cxx::_S_is_basic_char_type** (wchar_t ∗)
- template<class _CharT >
  bool **__gnu_cxx::_S_is_one_byte_char_type** (_CharT ∗)
- bool **__gnu_cxx::_S_is_one_byte_char_type** (char ∗)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator!=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator!=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator!=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)

- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator!=** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)
- template<class _CharT , class _Alloc >
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, const _CharT ∗__right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, const _CharT ∗__right)
- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT , class _Alloc >
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
  ptrdiff_t **__gnu_cxx::operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, ptrdiff_t __n)
- template<class _CharT , class _Alloc >
  ptrdiff_t **__gnu_cxx::operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator<** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator<** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator<** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT , class _Traits , class _Alloc >
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)

- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator<=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator<=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator<=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator==** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator==** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator==** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator==** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator>** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator>** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator>** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator>=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator>=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT , class _Alloc >
  bool **__gnu_cxx::operator>=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT , class __Alloc >
  void **__gnu_cxx::swap** (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)
- template<class _CharT , class _Alloc >
  void **__gnu_cxx::swap** (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)

Variables

- template<class _CharT , class _Alloc >
  rope< _CharT, _Alloc > **__gnu_cxx::identity_element** (_Rope_Concat_fn< _CharT, _Alloc >)

5.452.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file rope.

5.453 ropeimpl.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template< class _CharT , class _Traits >
  void **__gnu_cxx::_Rope_fill** (basic_ostream< _CharT, _Traits > &__o, size_t __n)
- template< class _CharT >
  bool **__gnu_cxx::_Rope_is_simple** (_CharT ∗)
- bool **__gnu_cxx::_Rope_is_simple** (char ∗)
- bool **__gnu_cxx::_Rope_is_simple** (wchar_t ∗)
- template< class _Rope_iterator >
  void **__gnu_cxx::_Rope_rotate** (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)
- template< class _CharT , class _Traits , class _Alloc >
  std::basic_ostream< _CharT,
  _Traits > & **__gnu_cxx::operator**<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)
- void **__gnu_cxx::rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)

5.453.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/rope>.

Definition in file ropeimpl.h.

5.454 rotate_fn_imps.hpp File Reference

5.454.1 Detailed Description

Contains imps for rotating nodes.

Definition in file bin_search_tree_/rotate_fn_imps.hpp.

5.455 rotate_fn_imps.hpp File Reference

5.455.1 Detailed Description

Contains imps for rotating nodes.

Definition in file pat_trie_/rotate_fn_imps.hpp.

5.456 safe_base.h File Reference

Classes

- class __gnu_debug::_Safe_iterator_base

Basic functionality for a safe iterator.

- class __gnu_debug::_Safe_sequence_base

  Base class that supports tracking of iterators that reference a sequence.

Namespaces

- namespace __gnu_debug

5.456.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_base.h.

5.457 safe_iterator.h File Reference

Classes

- struct __gnu_debug::_BeforeBeginHelper< _Sequence >
- class __gnu_debug::_Safe_iterator< _Iterator, _Sequence >

  Safe iterator wrapper.

Namespaces

- namespace __gnu_debug

Enumerations

- enum __gnu_debug::_Distance_precision { **__dp_equality**, **__dp_sign**, **__dp_exact** }

Functions

- bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base ∗__x)
- template<typename _Iterator1 , typename _Iterator2 >
  std::pair< typename
  std::iterator_traits
  < _Iterator1 >
  ::difference_type,
  _Distance_precision > __gnu_debug::__get_distance (const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::random_access_iterator_tag)
- template<typename _Iterator1 , typename _Iterator2 >
  std::pair< typename
  std::iterator_traits
  < _Iterator1 >
  ::difference_type,
  _Distance_precision > **__gnu_debug::__get_distance** (const _Iterator1 &__lhs, const _Iterator2 &__rhs, std::forward_iterator_tag)

- template<typename _Iterator1 , typename _Iterator2 >
std::pair< typename std::iterator_traits < _Iterator1 > ::difference_type, _Distance_precision > **__gnu_debug::__get_distance** (const _Iterator1 &__lhs, const _Iterator2 &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
bool **__gnu_debug::operator!=** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
bool **__gnu_debug::operator!=** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
_Safe_iterator< _Iterator, _Sequence > **__gnu_debug::operator+** (typename _Safe_iterator< _Iterator, _Sequence >::difference_type __n, const _Safe_iterator< _Iterator, _Sequence > &__i)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
_Safe_iterator< _IteratorL, _Sequence >::difference_type **__gnu_debug::operator-** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
_Safe_iterator< _Iterator, _Sequence >::difference_type **__gnu_debug::operator-** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
bool **__gnu_debug::operator<** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
bool **__gnu_debug::operator<** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
bool **__gnu_debug::operator<=** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
bool **__gnu_debug::operator<=** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
bool **__gnu_debug::operator==** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
bool **__gnu_debug::operator==** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
bool **__gnu_debug::operator>** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
bool **__gnu_debug::operator>** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
bool **__gnu_debug::operator>=** (const _Safe_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
bool **__gnu_debug::operator>=** (const _Safe_iterator< _Iterator, _Sequence > &__lhs, const _Safe_iterator< _Iterator, _Sequence > &__rhs)

5.457.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_iterator.h.

5.458 safe_iterator.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_ITERATOR_TCC**

5.458.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_iterator.tcc.

5.459 safe_local_iterator.h File Reference

Classes

- class __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >

  Safe iterator wrapper.

Namespaces

- namespace __gnu_debug

Functions

- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **__gnu_debug::operator!=** (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **__gnu_debug::operator!=** (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)
- template<typename _IteratorL , typename _IteratorR , typename _Sequence >
  bool **__gnu_debug::operator==** (const _Safe_local_iterator< _IteratorL, _Sequence > &__lhs, const _Safe_local_iterator< _IteratorR, _Sequence > &__rhs)
- template<typename _Iterator , typename _Sequence >
  bool **__gnu_debug::operator==** (const _Safe_local_iterator< _Iterator, _Sequence > &__lhs, const _Safe_local_iterator< _Iterator, _Sequence > &__rhs)

5.459.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_local_iterator.h.

5.460 safe_local_iterator.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_LOCAL_ITERATOR_TCC**

5.460.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_local_iterator.tcc.

5.461 safe_sequence.h File Reference

Classes

- class __gnu_debug::_After_nth_from< _Iterator >
- class __gnu_debug::_Equal_to< _Type >
- class __gnu_debug::_Not_equal_to< _Type >
- class __gnu_debug::_Safe_iterator< _Iterator, _Sequence >

  Safe iterator wrapper.
- class __gnu_debug::_Safe_sequence< _Sequence >

  Base class for constructing a safe sequence type that tracks iterators that reference it.

Namespaces

- namespace __gnu_debug

5.461.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_sequence.h.

5.462 safe_sequence.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_SEQUENCE_TCC**

5.462.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_sequence.tcc.

5.463 safe_unordered_base.h File Reference

Classes

- class __gnu_debug::_Safe_local_iterator_base
  Basic functionality for a safe iterator.
- class __gnu_debug::_Safe_unordered_container_base
  Base class that supports tracking of local iterators that reference an unordered container.

Namespaces

- namespace __gnu_debug

5.463.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_unordered_base.h.

5.464 safe_unordered_container.h File Reference

Classes

- class __gnu_debug::_Safe_unordered_container< _Container >
  Base class for constructing a safe unordered container type that tracks iterators that reference it.

Namespaces

- namespace __gnu_debug

5.464.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_unordered_container.h.

5.465 safe_unordered_container.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_UNORDERED_CONTAINER_TCC**

5.465.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file safe_unordered_container.tcc.

5.466 sample_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_probe_fn

  A sample probe policy.

Namespaces

- namespace __gnu_pbds

5.466.1 Detailed Description

Contains a sample probe policy.

Definition in file sample_probe_fn.hpp.

5.467 sample_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::sample_range_hashing

  A sample range-hashing functor.

Namespaces

- namespace __gnu_pbds

5.467.1 Detailed Description

Contains a range hashing policy.

Definition in file sample_range_hashing.hpp.

5.468 sample_ranged_hash_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_ranged_hash_fn

  A sample ranged-hash functor.

Namespaces

- namespace __gnu_pbds

5.468.1 Detailed Description

Contains a ranged hash policy.

Definition in file sample_ranged_hash_fn.hpp.

5.469 sample_ranged_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_ranged_probe_fn

  A sample ranged-probe functor.

Namespaces

- namespace __gnu_pbds

5.469.1 Detailed Description

Contains a ranged probe policy.

Definition in file sample_ranged_probe_fn.hpp.

5.470 sample_resize_policy.hpp File Reference

Classes

- class __gnu_pbds::sample_resize_policy

  A sample resize policy.

Namespaces

- namespace __gnu_pbds

5.470.1 Detailed Description

Contains a sample resize policy for hash tables.

Definition in file sample_resize_policy.hpp.

5.471 sample_resize_trigger.hpp File Reference

Classes

- class __gnu_pbds::sample_resize_trigger

  A sample resize trigger policy.

Namespaces

- namespace __gnu_pbds

5.471.1 Detailed Description

Contains a sample resize trigger policy class.

Definition in file sample_resize_trigger.hpp.

5.472 sample_size_policy.hpp File Reference

Classes

- class __gnu_pbds::sample_size_policy

  A sample size policy.

Namespaces

- namespace __gnu_pbds

5.472.1 Detailed Description

Contains a sample size resize-policy.

Definition in file sample_size_policy.hpp.

5.473 sample_tree_node_update.hpp File Reference

Classes

- class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >

  A sample node updator.

Namespaces

- namespace __gnu_pbds

5.473.1 Detailed Description

Contains a samle node update functor.

Definition in file sample_tree_node_update.hpp.

5.474 sample_trie_access_traits.hpp File Reference

Classes

- struct __gnu_pbds::sample_trie_access_traits

  A sample trie element access traits.

Namespaces

- namespace __gnu_pbds

5.474.1 Detailed Description

Contains a sample probe policy.

Definition in file sample_trie_access_traits.hpp.

5.475 sample_trie_node_update.hpp File Reference

Classes

- class __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

  A sample node updator.

Namespaces

- namespace __gnu_pbds

5.475.1 Detailed Description

Contains a samle node update functor.

Definition in file sample_trie_node_update.hpp.

5.476 sample_update_policy.hpp File Reference

Classes

- struct __gnu_pbds::sample_update_policy

  A sample list-update policy.

Namespaces

- namespace __gnu_pbds

5.476.1 Detailed Description

Contains a sample policy for list update containers.

Definition in file sample_update_policy.hpp.

5.477 scoped_allocator File Reference

Classes

- class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
  Primary class template.

Namespaces

- namespace std

Macros

- #define **_SCOPED_ALLOCATOR**

Functions

- template<typename _Alloc >
  auto **std::__do_outermost** (_Alloc &__a, _Alloc ∗) -> decltype(__a.outer_allocator())
- template<typename _Alloc >
  _Alloc & **std::__do_outermost** (_Alloc &__a,...)
- template<typename _Alloc >
  auto **std::__outermost** (_Alloc &__a) -> decltype(__do_outermost(__a,&__a))
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **std::operator!=** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept
- template<typename _OutA1 , typename _OutA2 , typename... _InA>
  bool **std::operator==** (const scoped_allocator_adaptor< _OutA1, _InA...> &__a, const scoped_allocator_adaptor< _OutA2, _InA...> &__b) noexcept

5.477.1 Detailed Description

This is a Standard C++ Library header.

Definition in file scoped_allocator.

5.478 search.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter , typename _DifferenceTp >
  void __gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp ∗__off)
- template<typename __RAIter1 , typename __RAIter2 , typename _Pred >
  __RAIter1 __gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)

5.478.1 Detailed Description

Parallel implementation base for std::search() and std::search_n(). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file search.h.

5.479 set File Reference

Macros

- #define **_GLIBCXX_SET**

5.479.1 Detailed Description

This is a Standard C++ Library header.

Definition in file set.

5.480 set File Reference

Macros

- #define **_GLIBCXX_DEBUG_SET**

5.480.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/set.

5.481 set File Reference

Macros

- #define **_GLIBCXX_PROFILE_SET**

5.481.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/set.

5.482 set.h File Reference

Classes

- class std::__debug::set< _Key, _Compare, _Allocator >

  Class std::set with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator!=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator<=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__debug::operator>=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **std::__debug::swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)

5.482.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/set.h.

5.483 set.h File Reference

Classes

- class std::__profile::set< _Key, _Compare, _Allocator >
  Class std::set wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Functions

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator!=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)

- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator<=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  bool **std::__profile::operator>=** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key , typename _Compare , typename _Allocator >
  void **std::__profile::swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)

5.483.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/set.h.

5.484 set_operations.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter , typename _OutputIterator >
  _OutputIterator **__gnu_parallel::__copy_tail** (std::pair< _IIter, _IIter > __b, std::pair< _IIter, _IIter > __e, _OutputIterator __r)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__gnu_parallel::__parallel_set_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__gnu_parallel::__parallel_set_intersection** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _Operation >
  _OutputIterator **__gnu_parallel::__parallel_set_operation** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Operation __op)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__gnu_parallel::__parallel_set_symmetric_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter , typename _OutputIterator , typename _Compare >
  _OutputIterator **__gnu_parallel::__parallel_set_union** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)

5.484.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file set_operations.h.

5.485 settings.h File Reference

Classes

- struct __gnu_parallel::_Settings

  class _Settings Run-time settings for the parallel mode including all tunable parameters.

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_PARALLEL_CONDITION(__c)

5.485.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms. This file is a GNU parallel extension to the Standard C++ Library.

5.485.2 parallelization_decision

The decision whether to run an algorithm in parallel.

There are several ways the user can switch on and __off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag gnu_parallel::sequential_tag() to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag
      ());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable __gnu_parallel::_Settings::algorithm_strategy is checked. _It is a tristate variable corresponding to:

a. force_sequential, meaning the sequential algorithm is executed. b. force_parallel, meaning the parallel algorithm is executed. c. heuristic

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(__s). The threshold can be set by the user, individually for each algorithm. The according variables are called gnu_parallel::_Settings::[algorithm]_minimal_n .

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

Definition in file settings.h.

5.485.3 Macro Definition Documentation

5.485.3.1 #define _GLIBCXX_PARALLEL_CONDITION(*__c*)

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

Parameters

__c	A condition that is convertible to bool that is overruled by __gnu_parallel::_Settings::algorithm_-strategy. Usually a decision based on the input size.

Definition at line 95 of file settings.h.

5.486 shared_ptr.h File Reference

Classes

- struct owner_less< _Tp >

  Primary template owner_less.
- class std::enable_shared_from_this< _Tp >

  Base class allowing use of member function shared_from_this.
- struct std::hash< shared_ptr< _Tp > >

  std::hash specialization for shared_ptr.
- struct std::owner_less< shared_ptr< _Tp > >

  Partial specialization of owner_less for shared_ptr.
- struct std::owner_less< weak_ptr< _Tp > >

  Partial specialization of owner_less for weak_ptr.
- class std::shared_ptr< _Tp >

  A smart pointer with reference-counted copy semantics.
- class std::weak_ptr< _Tp >

  A smart pointer with weak semantics.

Namespaces

- namespace std

Functions

- template<typename _Tp , typename _Alloc , typename... _Args>
  shared_ptr< _Tp > std::allocate_shared (const _Alloc &__a, _Args &&...__args)
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **std::const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **std::dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Del , typename _Tp , _Lock_policy _Lp>
  _Del ∗ std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Tp , typename... _Args>
  shared_ptr< _Tp > std::make_shared (_Args &&...__args)
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator<** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator<** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator<** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & std::operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 >
  bool **std::operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp >
  bool **std::operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp >
  bool **std::operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp , typename _Tp1 >
  shared_ptr< _Tp > **std::static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept

- template<typename _Tp >
  void **std::swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp >
  void **std::swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept

5.486.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file shared_ptr.h.

5.487 shared_ptr_base.h File Reference

Classes

- class std::bad_weak_ptr

  Exception possibly thrown by `shared_ptr`.
- class std::enable_shared_from_this< _Tp >

  Base class allowing use of member function shared_from_this.
- struct std::hash< __shared_ptr< _Tp, _Lp > >

  std::hash specialization for __shared_ptr.
- class std::shared_ptr< _Tp >

  A smart pointer with reference-counted copy semantics.
- class std::weak_ptr< _Tp >

  A smart pointer with weak semantics.

Namespaces

- namespace std

Functions

- template<typename _Tp , _Lock_policy _Lp, typename _Alloc , typename... _Args>
  __shared_ptr< _Tp, _Lp > **std::__allocate_shared** (const _Alloc &__a, _Args &&...__args)
- template<_Lock_policy _Lp, typename _Tp1 , typename _Tp2 >
  void **std::__enable_shared_from_this_helper** (const __shared_count< _Lp > &, const __enable_shared_from_this< _Tp1, _Lp > ∗, const _Tp2 ∗) noexcept
- template<typename _Tp1 , typename _Tp2 >
  void **std::__enable_shared_from_this_helper** (const __shared_count<> &, const enable_shared_from_this< _Tp1 > ∗, const _Tp2 ∗) noexcept
- template<_Lock_policy _Lp>
  void **std::__enable_shared_from_this_helper** (const __shared_count< _Lp > &,...) noexcept
- template<typename _Tp , _Lock_policy _Lp, typename... _Args>
  __shared_ptr< _Tp, _Lp > **std::__make_shared** (_Args &&...__args)
- void **std::__throw_bad_weak_ptr** ()
- template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept

- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **std::operator!=** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator!=** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator!=** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **std::operator<** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator<** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator<** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **std::operator<=** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator<=** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator<=** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **std::operator==** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator==** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator==** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **std::operator>** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator>** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator>** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>
  bool **std::operator>=** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator>=** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  bool **std::operator>=** (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept
- template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  void **std::swap** (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Tp , _Lock_policy _Lp>
  void **std::swap** (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept

5.487.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file shared_ptr_base.h.

5.488 size_fn_imps.hpp File Reference

5.488.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container size related functions.

Definition in file size_fn_imps.hpp.

5.489 slice_array.h File Reference

Classes

- class std::slice

  Class defining one-dimensional subset of an array.
- class std::slice_array< _Tp >

  Reference to one-dimensional subset of an array.

Namespaces

- namespace std

Macros

- #define **_DEFINE_VALARRAY_OPERATOR**(_Op, _Name)

5.489.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file slice_array.h.

5.490 slist File Reference

Classes

- class __gnu_cxx::slist< _Tp, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_SLIST**

Functions

- _Slist_node_base ∗ **__gnu_cxx::__slist_make_link** (_Slist_node_base ∗__prev_node, _Slist_node_base ∗__new_node)
- _Slist_node_base ∗ **__gnu_cxx::__slist_previous** (_Slist_node_base ∗__head, const _Slist_node_base ∗__node)
- const _Slist_node_base ∗ **__gnu_cxx::__slist_previous** (const _Slist_node_base ∗__head, const _Slist_node_base ∗__node)
- _Slist_node_base ∗ **__gnu_cxx::__slist_reverse** (_Slist_node_base ∗__node)
- size_t **__gnu_cxx::__slist_size** (_Slist_node_base ∗__node)
- void **__gnu_cxx::__slist_splice_after** (_Slist_node_base ∗__pos, _Slist_node_base ∗__before_first, _Slist_node_base ∗__before_last)
- void **__gnu_cxx::__slist_splice_after** (_Slist_node_base ∗__pos, _Slist_node_base ∗__head)
- template<class _Tp , class _Alloc >
  bool **__gnu_cxx::operator!=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp , class _Alloc >
  bool **__gnu_cxx::operator<** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp , class _Alloc >
  bool **__gnu_cxx::operator<=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp , class _Alloc >
  bool **__gnu_cxx::operator==** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp , class _Alloc >
  bool **__gnu_cxx::operator>** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp , class _Alloc >
  bool **__gnu_cxx::operator>=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp , class _Alloc >
  void **__gnu_cxx::swap** (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)

5.490.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

Definition in file slist.

5.491 sort.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<bool __stable, typename _RAIter , typename _Compare , typename _Parallelism >
  void **__gnu_parallel::__parallel_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)

- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)
- template<bool __stable, typename _RAIter , typename _Compare >
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)

5.491.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file sort.h.

5.492 splay_fn_imps.hpp File Reference

5.492.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_fn_imps.hpp.

5.493 splay_tree_.hpp File Reference

Classes

- class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
  Splay tree.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node)
- #define **PB_DS_CLASS_C_DEC**

- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_S_TREE_BASE**
- #define **PB_DS_S_TREE_BASE_NAME**
- #define **PB_DS_S_TREE_NAME**

5.493.1 Detailed Description

Contains an implementation class for splay trees.

Definition in file splay_tree_.hpp.

5.494 split_fn_imps.hpp File Reference

5.494.1 Detailed Description

Contains an implementation class for pat_trie.

Definition in file split_fn_imps.hpp.

5.495 split_join_fn_imps.hpp File Reference

5.495.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/split_join_fn_imps.hpp.

5.496 split_join_fn_imps.hpp File Reference

5.496.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

Definition in file binomial_heap_base_/split_join_fn_imps.hpp.

5.497 split_join_fn_imps.hpp File Reference

5.497.1 Detailed Description

Contains an implementation class for bin_search_tree_.

Definition in file bin_search_tree_/split_join_fn_imps.hpp.

5.498 split_join_fn_imps.hpp File Reference

5.498.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/split_join_fn_imps.hpp.

5.499 split_join_fn_imps.hpp File Reference

5.499.1 Detailed Description

Contains an implementation class for a pairing heap.

Definition in file pairing_heap_/split_join_fn_imps.hpp.

5.500 split_join_fn_imps.hpp File Reference

5.500.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/split_join_fn_imps.hpp.

5.501 split_join_fn_imps.hpp File Reference

5.501.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

Definition in file rc_binomial_heap_/split_join_fn_imps.hpp.

5.502 split_join_fn_imps.hpp File Reference

5.502.1 Detailed Description

Contains an implementation class for splay_tree_.

Definition in file splay_tree_/split_join_fn_imps.hpp.

5.503 split_join_fn_imps.hpp File Reference

5.503.1 Detailed Description

Contains an implementation for thin_heap_.

Definition in file thin_heap_/split_join_fn_imps.hpp.

5.504 sso_string_base.h File Reference

Namespaces

- namespace __gnu_cxx

5.504.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

Definition in file sso_string_base.h.

5.505 sstream File Reference

Classes

- class std::basic_istringstream< _CharT, _Traits, _Alloc >

  Controlling input for std::string.
- class std::basic_ostringstream< _CharT, _Traits, _Alloc >

  Controlling output for std::string.
- class std::basic_stringbuf< _CharT, _Traits, _Alloc >

  The actual work of input and output (for std::string).
- class std::basic_stringstream< _CharT, _Traits, _Alloc >

  Controlling input and output for std::string.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_SSTREAM**

5.505.1 Detailed Description

This is a Standard C++ Library header.

Definition in file sstream.

5.506 sstream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_SSTREAM_TCC**

5.506.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <sstream>.

Definition in file sstream.tcc.

5.507 stack File Reference

Macros

- #define **_GLIBCXX_STACK**

5.507.1 Detailed Description

This is a Standard C++ Library header.

Definition in file stack.

5.508 standard_policies.hpp File Reference

Classes

- struct __gnu_pbds::detail::default_comb_hash_fn

  Primary template, default_comb_hash_fn.
- struct __gnu_pbds::detail::default_eq_fn< Key >

  Primary template, default_eq_fn.
- struct __gnu_pbds::detail::default_hash_fn< Key >

  Primary template, default_hash_fn.
- struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >

  Primary template, default_probe_fn.
- struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >

  Primary template, default_resize_policy.
- struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >

  Partial specialization, default_trie_access_traits.
- struct __gnu_pbds::detail::default_update_policy

  Default update policy.
- struct default_trie_access_traits< Key >

  Primary template, default_trie_access_traits.

Namespaces

- namespace __gnu_pbds

Macros

- #define **__dtrie_alloc**
- #define **__dtrie_string**

Enumerations

- enum { **default_store_hash** }

5.508.1 Detailed Description

Contains standard policies for containers.

Definition in file standard_policies.hpp.

5.509 stdc++.h File Reference

5.509.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file stdc++.h.

5.510 stdexcept File Reference

Classes

- class std::domain_error
- class std::invalid_argument
- class std::length_error
- class std::logic_error

  One of two subclasses of exception.
- class std::out_of_range
- class std::overflow_error
- class std::range_error
- class std::runtime_error

  One of two subclasses of exception.
- class std::underflow_error

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_STDEXCEPT**

5.510.1 Detailed Description

This is a Standard C++ Library header.

Definition in file stdexcept.

5.511 stdio_filebuf.h File Reference

Classes

- class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

  Provides a layer of compatibility for C/POSIX.
  This GNU extension provides extensions for working with standard C FILE∗'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

Namespaces

- namespace __gnu_cxx

5.511.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file stdio_filebuf.h.

5.512 stdio_sync_filebuf.h File Reference

Classes

- class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

  Provides a layer of compatibility for C.
  This GNU extension provides extensions for working with standard C FILE∗'s. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

Namespaces

- namespace __gnu_cxx

5.512.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file stdio_sync_filebuf.h.

5.513 stdtr1c++.h File Reference

5.513.1 Detailed Description

This is an implementation file for a precompiled header.

Definition in file stdtr1c++.h.

5.514 stl_algo.h File Reference

Namespaces

- namespace std

Enumerations

- enum { **_S_threshold** }
- enum { **_S_chunk_size** }

Functions

- template<typename _RandomAccessIterator , typename _Distance >
  void **std::__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size)

- template<typename _RandomAccessIterator , typename _Distance , typename _Compare >
  void **std::__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  _OutputIterator **std::__copy_n** (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Size , typename _OutputIterator >
  _OutputIterator **std::__copy_n** (_RandomAccessIterator __first, _Size __n, _OutputIterator __result, random_access_iterator_tag)
- template<typename _RandomAccessIterator >
  void std::__final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::__final_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _Tp >
  _InputIterator std::__find (_InputIterator __first, _InputIterator __last, const _Tp &__val, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Tp >
  _RandomAccessIterator std::__find (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__val, random_access_iterator_tag)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 **std::__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 **std::__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 >
  _BidirectionalIterator1 **std::__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BinaryPredicate >
  _BidirectionalIterator1 **std::__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::__find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Predicate >
  _RandomAccessIterator std::__find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::__find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Predicate >
  _RandomAccessIterator std::__find_if_not (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::__find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Predicate , typename _Distance >
  _InputIterator std::__find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)
- template<typename _EuclideanRingElement >
  _EuclideanRingElement std::__gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)
- template<typename _RandomAccessIterator >
  void std::__heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::__heap_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)

- template<typename _ForwardIterator , typename _Predicate , typename _Distance >
  _ForwardIterator std::__inplace_stable_partition (_ForwardIterator __first, _Predicate __pred, _Distance __len)
- template<typename _RandomAccessIterator >
  void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::__insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Size >
  void **std::__introselect** (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit)
- template<typename _RandomAccessIterator , typename _Size , typename _Compare >
  void **std::__introselect** (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Size >
  void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit)
- template<typename _RandomAccessIterator , typename _Size , typename _Compare >
  void std::__introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)
- template<typename _BidirectionalIterator , typename _Distance , typename _Pointer >
  void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size)
- template<typename _BidirectionalIterator , typename _Distance , typename _Pointer , typename _Compare >
  void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance >
  void **std::__merge_sort_loop** (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size)
- template<typename _RandomAccessIterator1 , typename _RandomAccessIterator2 , typename _Distance , typename _Compare >
  void **std::__merge_sort_loop** (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Pointer >
  void **std::__merge_sort_with_buffer** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer)
- template<typename _RandomAccessIterator , typename _Pointer , typename _Compare >
  void **std::__merge_sort_with_buffer** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)
- template<typename _BidirectionalIterator , typename _Distance >
  void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2)
- template<typename _BidirectionalIterator , typename _Distance , typename _Compare >
  void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)
- template<typename _Iterator >
  void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c)
- template<typename _Iterator , typename _Compare >
  void std::__move_median_first (_Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)

- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::__move_merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 >
  void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _BidirectionalIterator3 , typename _Compare >
  void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)
- template<typename _BidirectionalIterator , typename _Predicate >
  _BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)
- template<typename _BidirectionalIterator >
  void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator >
  void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _ForwardIterator >
  void std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)
- template<typename _BidirectionalIterator >
  void std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator >
  void std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance >
  _BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)
- template<typename _ForwardIterator , typename _Integer , typename _Tp >
  _ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, std::forward_iterator_tag)
- template<typename _RandomAccessIter , typename _Integer , typename _Tp >
  _RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, std::random_access_iterator_tag)
- template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _ForwardIterator std::__search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::forward_iterator_tag)
- template<typename _RandomAccessIter , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _RandomAccessIter std::__search_n (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, std::random_access_iterator_tag)
- template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance >
  _ForwardIterator std::__stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)

- template<typename _RandomAccessIterator , typename _Pointer , typename _Distance >
  void **std::__stable_sort_adaptive** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size)
- template<typename _RandomAccessIterator , typename _Pointer , typename _Distance , typename _Compare >
  void **std::__stable_sort_adaptive** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::__unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::__unguarded_linear_insert (_RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::__unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Tp >
  _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot)
- template<typename _RandomAccessIterator , typename _Tp , typename _Compare >
  _RandomAccessIterator std::__unguarded_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, const _Tp &__pivot, _Compare __comp)
- template<typename _RandomAccessIterator >
  _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator std::__unguarded_partition_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _ForwardIterator , typename _OutputIterator >
  _OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, forward_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, input_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _ForwardIterator >
  _ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, input_iterator_tag, forward_iterator_tag)
- template<typename _ForwardIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator std::__unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)
- template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator std::__unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)
- template<typename _ForwardIterator >
  _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator , typename _Predicate >
  bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)

- template<typename _InputIterator , typename _Predicate >
  bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator , typename _Tp >
  bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
  _OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _InputIterator , typename _Size , typename _OutputIterator >
  _OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _InputIterator , typename _Tp >
  iterator_traits
  < _InputIterator >
  ::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _InputIterator , typename _Predicate >
  iterator_traits
  < _InputIterator >
  ::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator , typename _Tp >
  pair< _ForwardIterator,
  _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  pair< _ForwardIterator,
  _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _InputIterator , typename _Tp >
  _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _InputIterator , typename _ForwardIterator >
  _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)
- template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
  _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Predicate >
  _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _Function >
  _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _ForwardIterator , typename _Generator >
  void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _OutputIterator , typename _Size , typename _Generator >
  _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)
- template<typename _InputIterator1 , typename _InputIterator2 >
  bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)

- template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare >
  bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)
- template<typename _BidirectionalIterator >
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _Predicate >
  bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator >
  bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator >
  _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp >
  _Tp **std::max** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  _Tp **std::max** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator >
  _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp >
  _Tp **std::min** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  _Tp **std::min** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator >
  _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp >
  pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)

- template<typename _Tp >
  pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >)
- template<typename _Tp , typename _Compare >
  pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator >
  pair< _ForwardIterator,
  _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Compare >
  pair< _ForwardIterator,
  _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _BidirectionalIterator >
  bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _Predicate >
  bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator >
  void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _RandomAccessIterator >
  _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator , typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename _Predicate >
  pair< _OutputIterator1,
  _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _BidirectionalIterator >
  bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _Compare >
  bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _RandomNumberGenerator >
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
  _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator , typename _Tp >
  void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator , typename _OutputIterator , typename _Tp >
  _OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp >
  _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _ForwardIterator , typename _Predicate , typename _Tp >
  void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- template<typename _BidirectionalIterator >
  void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator , typename _OutputIterator >
  _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- template<typename _ForwardIterator >
  void std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _OutputIterator >
  _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
  _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _ForwardIterator , typename _Integer , typename _Tp >
  _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator , typename _Integer , typename _Tp , typename _BinaryPredicate >
  _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)

- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
  _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _Compare >
  _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _RandomAccessIterator , typename _UniformRandomNumberGenerator >
  void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)
- template<typename _RandomAccessIterator >
  void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _ForwardIterator , typename _Predicate >
  _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator >
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >
  _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _ForwardIterator >
  _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _BinaryPredicate >
  _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator , typename _OutputIterator >
  _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
  _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator , typename _Tp , typename _Compare >
  _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)

5.514.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

Definition in file stl_algo.h.

5.515 stl_algobase.h File Reference

Classes

- struct std::char_traits< _CharT >

  Basis for explicit traits specializations.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOVE3**(_Tp, _Up, _Vp)
- #define **_GLIBCXX_MOVE_BACKWARD3**(_Tp, _Up, _Vp)

Functions

- template<bool _IsMove, typename _II , typename _OI >
  _OI **std::__copy_move_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT,
  char_traits< _CharT >
  > >::__type **std::__copy_move_a2** (_CharT ∗, _CharT ∗, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT,
  char_traits< _CharT >
  > >::__type **std::__copy_move_a2** (const _CharT ∗, const _CharT ∗, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  _CharT ∗ >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT ∗)
- template<bool _IsMove, typename _II , typename _OI >
  _OI **std::__copy_move_a2** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _BI1 , typename _BI2 >
  _BI2 **std::__copy_move_backward_a** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _BI1 , typename _BI2 >
  _BI2 **std::__copy_move_backward_a2** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _II1 , typename _II2 >
  bool **std::__equal_aux** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _ForwardIterator , typename _Tp >
  __gnu_cxx::__enable_if
  <!__is_scalar< _Tp >::__value,
  void >::__type **std::__fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- template<typename _Tp >
  __gnu_cxx::__enable_if
  < __is_byte< _Tp >::__value,
  void >::__type **std::__fill_a** (_Tp ∗__first, _Tp ∗__last, const _Tp &__c)
- template<typename _OutputIterator , typename _Size , typename _Tp >
  __gnu_cxx::__enable_if
  <!__is_scalar< _Tp >::__value,
  _OutputIterator >::__type **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value)
- template<typename _Size , typename _Tp >
  __gnu_cxx::__enable_if
  < __is_byte< _Tp >::__value,
  _Tp ∗ >::__type **std::__fill_n_a** (_Tp ∗__first, _Size __n, const _Tp &__c)
- template<typename _II1 , typename _II2 >
  bool **std::__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- constexpr int std::__lg (int __n)
- constexpr unsigned **std::__lg** (unsigned __n)
- constexpr long **std::__lg** (long __n)
- constexpr unsigned long **std::__lg** (unsigned long __n)
- constexpr long long **std::__lg** (long long __n)
- constexpr unsigned long long **std::__lg** (unsigned long long __n)
- template<typename _Iterator >
  _Miter_base< _Iterator >
  ::iterator_type **std::__miter_base** (_Iterator __it)
- template<typename _Iterator >
  _Niter_base< _Iterator >
  ::iterator_type **std::__niter_base** (_Iterator __it)
- template<typename _II , typename _OI >
  _OI std::copy (_II __first, _II __last, _OI __result)
- template<typename _BI1 , typename _BI2 >
  _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
  bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _II1 , typename _II2 >
  bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _ForwardIterator , typename _Tp >
  void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI , typename _Size , typename _Tp >
  _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
  void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _II1 , typename _II2 >
  bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1 , typename _II2 , typename _Compare >
  bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _ForwardIterator , typename _Tp >
  _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _Tp >
  const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp , typename _Compare >
  const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp >
  const _Tp & std::min (const _Tp &__a, const _Tp &__b)

- template<typename _Tp , typename _Compare >
const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _InputIterator1 , typename _InputIterator2 >
pair< _InputIterator1,
_InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
pair< _InputIterator1,
_InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _II , typename _OI >
_OI std::move (_II __first, _II __last, _OI __result)
- template<typename _BI1 , typename _BI2 >
_BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _ForwardIterator1 , typename _ForwardIterator2 >
_ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)

5.515.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

Definition in file stl_algobase.h.

5.516 stl_bvector.h File Reference

Classes

- struct std::hash<::vector< bool, _Alloc > >

  std::hash specialization for vector<bool>.
- class std::vector< bool, _Alloc >

  A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Namespaces

- namespace std

Typedefs

- typedef unsigned long **std::_Bit_type**

Enumerations

- enum { **_S_word_bit** }

Functions

- void **std::__fill_bvector** (_Bit_iterator __first, _Bit_iterator __last, bool __x)
- void **std::fill** (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)

- _Bit_iterator **std::operator+** (ptrdiff_t __n, const _Bit_iterator &__x)
- _Bit_const_iterator **std::operator+** (ptrdiff_t __n, const _Bit_const_iterator &__x)
- ptrdiff_t **std::operator-** (const _Bit_iterator_base &__x, const _Bit_iterator_base &__y)
- void **std::swap** (_Bit_reference __x, _Bit_reference __y) noexcept
- void **std::swap** (_Bit_reference __x, bool &__y) noexcept
- void **std::swap** (bool &__x, _Bit_reference __y) noexcept

5.516.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

Definition in file stl_bvector.h.

5.517 stl_construct.h File Reference

Classes

- struct std::allocator< _Tp >

  The standard allocator, as per [20.4].

Namespaces

- namespace std

Functions

- template<typename _T1 , typename... _Args>
  void std::_Construct (_T1 ∗__p, _Args &&...__args)
- template<typename _Tp >
  void std::_Destroy (_Tp ∗__pointer)
- template<typename _ForwardIterator >
  void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Allocator >
  void **std::_Destroy** (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Tp >
  void **std::_Destroy** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)

5.517.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file stl_construct.h.

5.518 stl_deque.h File Reference

Classes

- class std::_Deque_base< _Tp, _Alloc >

- struct std::_Deque_iterator< _Tp, _Ref, _Ptr >

  A deque::iterator.
- class std::deque< _Tp, _Alloc >

  A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Namespaces

- namespace std

Macros

- #define _GLIBCXX_DEQUE_BUF_SIZE

Functions

- size_t **std::__deque_buf_size** (size_t __size)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::copy** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::copy** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::copy_backward** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::copy_backward** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  void **std::fill** (const _Deque_iterator< _Tp, _Tp &, _Tp ∗ > &__first, const _Deque_iterator< _Tp, _Tp &, _Tp ∗ > &__last, const _Tp &__value)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::move** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::move** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::move_backward** (_Deque_iterator< _Tp, _Tp &, _Tp ∗ > __first, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)
- template<typename _Tp >
  _Deque_iterator< _Tp, _Tp &, _Tp ∗ > **std::move_backward** (_Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __first, _Deque_iterator< _Tp, const _Tp &, const _Tp ∗ > __last, _Deque_iterator< _Tp, _Tp &, _Tp ∗ > __result)

- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **std::operator!=** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **std::operator!=** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  _Deque_iterator< _Tp, _Ref, _Ptr > **std::operator+** (ptrdiff_t __n, const _Deque_iterator< _Tp, _Ref, _Ptr > &__x)
- template<typename _Tp , typename _Ref , typename _Ptr >
  _Deque_iterator< _Tp, _Ref,
  _Ptr >::difference_type **std::operator-** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  _Deque_iterator< _Tp, _RefL,
  _PtrL >::difference_type **std::operator-** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **std::operator<** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **std::operator<** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **std::operator<=** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **std::operator<=** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **std::operator==** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **std::operator==** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **std::operator>** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)
- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **std::operator>** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Ref , typename _Ptr >
  bool **std::operator>=** (const _Deque_iterator< _Tp, _Ref, _Ptr > &__x, const _Deque_iterator< _Tp, _Ref, _Ptr > &__y)

- template<typename _Tp , typename _RefL , typename _PtrL , typename _RefR , typename _PtrR >
  bool **std::operator>=** (const _Deque_iterator< _Tp, _RefL, _PtrL > &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator>= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y)

5.518.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <deque>.

Definition in file stl_deque.h.

5.518.2 Macro Definition Documentation

5.518.2.1 #define _GLIBCXX_DEQUE_BUF_SIZE

This function controls the size of memory nodes.

Parameters

__size	The size of an element.

Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch _GLIBCXX_DEQUE_BUF_SIZE only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 85 of file stl_deque.h.

5.519 stl_function.h File Reference

Classes

- struct std::binary_function< _Arg1, _Arg2, _Result >
- class std::binary_negate< _Predicate >

  One of the negation functors.
- class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::const_mem_fun1_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::const_mem_fun_ref_t< _Ret, _Tp >

  One of the adaptors for member pointers.
- class std::const_mem_fun_t< _Ret, _Tp >

  One of the adaptors for member pointers.

- struct std::divides< _Tp >

  One of the math functors.
- struct std::equal_to< _Tp >

  One of the comparison functors.
- struct std::greater< _Tp >

  One of the comparison functors.
- struct std::greater_equal< _Tp >

  One of the comparison functors.
- struct std::less< _Tp >

  One of the comparison functors.
- struct std::less_equal< _Tp >

  One of the comparison functors.
- struct std::logical_and< _Tp >

  One of the Boolean operations functors.
- struct std::logical_not< _Tp >

  One of the Boolean operations functors.
- struct std::logical_or< _Tp >

  One of the Boolean operations functors.
- class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::mem_fun1_t< _Ret, _Tp, _Arg >

  One of the adaptors for member pointers.
- class std::mem_fun_ref_t< _Ret, _Tp >

  One of the adaptors for member pointers.
- class std::mem_fun_t< _Ret, _Tp >

  One of the adaptors for member pointers.
- struct std::minus< _Tp >

  One of the math functors.
- struct std::modulus< _Tp >

  One of the math functors.
- struct std::multiplies< _Tp >

  One of the math functors.
- struct std::negate< _Tp >

  One of the math functors.
- struct std::not_equal_to< _Tp >

  One of the comparison functors.
- struct std::plus< _Tp >

  One of the math functors.
- class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >

  One of the adaptors for function pointers.
- class std::pointer_to_unary_function< _Arg, _Result >

  One of the adaptors for function pointers.
- struct std::unary_function< _Arg, _Result >
- class std::unary_negate< _Predicate >

  One of the negation functors.

Namespaces

- namespace std

Functions

- template<typename _Ret , typename _Tp >
  mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::∗__f)())
- template<typename _Ret , typename _Tp , typename _Arg >
  mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::∗__f)(_Arg))
- template<typename _Ret , typename _Tp >
  mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::∗__f)())
- template<typename _Ret , typename _Tp , typename _Arg >
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::∗__f)(_Arg))
- template<typename _Predicate >
  unary_negate< _Predicate > std::not1 (const _Predicate &__pred)
- template<typename _Predicate >
  binary_negate< _Predicate > std::not2 (const _Predicate &__pred)
- template<typename _Arg , typename _Result >
  pointer_to_unary_function
  < _Arg, _Result > std::ptr_fun (_Result(∗__x)(_Arg))
- template<typename _Arg1 , typename _Arg2 , typename _Result >
  pointer_to_binary_function
  < _Arg1, _Arg2, _Result > std::ptr_fun (_Result(∗__x)(_Arg1, _Arg2))

5.519.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

Definition in file stl_function.h.

5.520 stl_heap.h File Reference

Namespaces

- namespace std

Functions

- template<typename _RandomAccessIterator , typename _Distance , typename _Tp >
  void **std::__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value)
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp , typename _Compare >
  void **std::__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Distance >
  bool **std::__is_heap** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator , typename _Compare , typename _Distance >
  bool **std::__is_heap** (_RandomAccessIterator __first, _Compare __comp, _Distance __n)

- template<typename _RandomAccessIterator >
  bool **std::__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  bool **std::__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Distance >
  _Distance **std::__is_heap_until** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator , typename _Distance , typename _Compare >
  _Distance **std::__is_heap_until** (_RandomAccessIterator __first, _Distance __n, _Compare __comp)
- template<typename _RandomAccessIterator >
  void **std::__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result)
- template<typename _RandomAccessIterator , typename _Compare >
  void **std::__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare __comp)
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp >
  void **std::__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value)
- template<typename _RandomAccessIterator , typename _Distance , typename _Tp , typename _Compare >
  void **std::__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare __comp)
- template<typename _RandomAccessIterator >
  bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator >
  void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator , typename _Compare >
  void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)

5.520.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <queue>.

Definition in file stl_heap.h.

5.521 stl_iterator.h File Reference

Classes

- class std::back_insert_iterator< _Container >

  Turns assignment into insertion.
- class std::front_insert_iterator< _Container >

  Turns assignment into insertion.
- class std::insert_iterator< _Container >

  Turns assignment into insertion.
- class std::move_iterator< _Iterator >
- class std::reverse_iterator< _Iterator >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR**(_Iter)
- #define **_GLIBCXX_MAKE_MOVE_ITERATOR**(_Iter)

Functions

- template<typename _Iterator , typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>
  _ReturnType **std::__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Container >
  back_insert_iterator< _Container > std::back_inserter (_Container &__x)
- template<typename _Container >
  front_insert_iterator< _Container > std::front_inserter (_Container &__x)
- template<typename _Container , typename _Iterator >
  insert_iterator< _Container > std::inserter (_Container &__x, _Iterator __i)
- template<typename _Iterator >
  move_iterator< _Iterator > **std::make_move_iterator** (_Iterator __i)
- template<typename _Iterator >
  bool **std::operator!=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **__gnu_cxx::operator!=** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **__gnu_cxx::operator!=** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator!=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator!=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)

- template<typename _Iterator >
  reverse_iterator< _Iterator > **std::operator+** (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _Iterator , typename _Container >
  __normal_iterator< _Iterator,
  _Container > **__gnu_cxx::operator+** (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i)
- template<typename _Iterator >
  move_iterator< _Iterator > **std::operator+** (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator >
  reverse_iterator< _Iterator >
  ::difference_type **std::operator-** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  auto **std::operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base()-__x.base())
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  auto **__gnu_cxx::operator-** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) -> decltype(__lhs.base()-__rhs.base())
- template<typename _Iterator , typename _Container >
  __normal_iterator< _Iterator,
  _Container >::difference_type **__gnu_cxx::operator-** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  auto **std::operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(__x.base()-__y.base())
- template<typename _Iterator >
  auto **std::operator-** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y) -> decltype(__x.base()-__y.base())
- template<typename _Iterator >
  bool **std::operator<** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **__gnu_cxx::operator<** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **__gnu_cxx::operator<** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator<** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  bool **std::operator<=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **__gnu_cxx::operator<=** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **__gnu_cxx::operator<=** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)

- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator<=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator<=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator==** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **__gnu_cxx::operator==** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **__gnu_cxx::operator==** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  bool **std::operator>** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **__gnu_cxx::operator>** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **__gnu_cxx::operator>** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator>** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _Iterator >
  bool **std::operator>=** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL , typename _IteratorR , typename _Container >
  bool **__gnu_cxx::operator>=** (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs)
- template<typename _Iterator , typename _Container >
  bool **__gnu_cxx::operator>=** (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs)
- template<typename _IteratorL , typename _IteratorR >
  bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator >
  bool **std::operator>=** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)

5.521.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

This file implements reverse_iterator, back_insert_iterator, front_insert_iterator, insert_iterator, __normal_iterator, and their supporting functions and overloaded operators.

Definition in file stl_iterator.h.

5.522 stl_iterator_base_funcs.h File Reference

Namespaces

- namespace std

Functions

- template<typename _InputIterator , typename _Distance >
  void **std::__advance** (_InputIterator &__i, _Distance __n, input_iterator_tag)
- template<typename _BidirectionalIterator , typename _Distance >
  void **std::__advance** (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator , typename _Distance >
  void **std::__advance** (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)
- template<typename _InputIterator >
  iterator_traits
  < _InputIterator >
  ::difference_type **std::__distance** (_InputIterator __first, _InputIterator __last, input_iterator_tag)
- template<typename _RandomAccessIterator >
  iterator_traits
  < _RandomAccessIterator >
  ::difference_type **std::__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _InputIterator , typename _Distance >
  void std::advance (_InputIterator &__i, _Distance __n)
- template<typename _InputIterator >
  iterator_traits
  < _InputIterator >
  ::difference_type std::distance (_InputIterator __first, _InputIterator __last)
- template<typename _ForwardIterator >
  _ForwardIterator **std::next** (_ForwardIterator __x, typename iterator_traits< _ForwardIterator >::difference_type __n=1)
- template<typename _BidirectionalIterator >
  _BidirectionalIterator **std::prev** (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)

5.522.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

This file contains all of the general iterator-related utility functions, such as distance() and advance().

Definition in file stl_iterator_base_funcs.h.

5.523 stl_iterator_base_types.h File Reference

Classes

- class std::__has_iterator_category_helper< _Tp >

  Traits class for iterators.
- struct std::bidirectional_iterator_tag

  Bidirectional iterators support a superset of forward iterator operations.
- struct std::forward_iterator_tag

  Forward iterators support a superset of input iterator operations.
- struct std::input_iterator_tag

  Marking input iterators.
- struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >

  Common iterator class.
- struct std::iterator_traits< _Tp * >

  Partial specialization for pointer types.
- struct std::iterator_traits< const _Tp * >

  Partial specialization for const pointer types.
- struct std::output_iterator_tag

  Marking output iterators.
- struct std::random_access_iterator_tag

  Random-access iterators support a superset of bidirectional iterator operations.

Namespaces

- namespace std

Typedefs

- template<typename _InIter >
  using **std::_RequireInputIter** = typename enable_if< is_convertible< typename iterator_traits< _InIter >::iterator_category, input_iterator_tag >::value >::type

Functions

- template<typename _Iter >
  iterator_traits< _Iter >
  ::iterator_category std::__iterator_category (const _Iter &)

5.523.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

This file contains all of the general iterator-related utility types, such as iterator_traits and struct iterator.

Definition in file stl_iterator_base_types.h.

5.524 stl_list.h File Reference

Classes

- struct std::__detail::_List_node_base

  Common part of a node in the list.
- class std::_List_base< _Tp, _Alloc >

  See bits/stl_deque.h's _Deque_base for an explanation.
- struct std::_List_const_iterator< _Tp >

  A list::const_iterator.
- struct std::_List_iterator< _Tp >

  A list::iterator.
- struct std::_List_node< _Tp >

  An actual node in the list.
- class std::list< _Tp, _Alloc >

  A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Val >
  bool **std::operator!=** (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Val >
  bool **std::operator==** (const _List_iterator< _Val > &__x, const _List_const_iterator< _Val > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator>= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y)

5.524.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <list>.

Definition in file stl_list.h.

5.525 stl_map.h File Reference

Classes

- class std::map< _Key, _Tp, _Compare, _Alloc >

  A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Namespaces

- namespace std

Functions

- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)

5.525.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map>.

Definition in file stl_map.h.

5.526 stl_multimap.h File Reference

Classes

- class std::multimap< _Key, _Tp, _Compare, _Alloc >

  A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Namespaces

- namespace std

Functions

- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
  void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y)

5.526.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map>.

Definition in file stl_multimap.h.

5.527 stl_multiset.h File Reference

Classes

- class std::multiset< _Key, _Compare, _Alloc >

  A standard container made up of elements, which can be retrieved in logarithmic time.

Namespaces

- namespace std

Functions

- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)

- template< typename _Key , typename _Compare , typename _Alloc >
  bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template< typename _Key , typename _Compare , typename _Alloc >
  bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template< typename _Key , typename _Compare , typename _Alloc >
  bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template< typename _Key , typename _Compare , typename _Alloc >
  bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template< typename _Key , typename _Compare , typename _Alloc >
  void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y)

5.527.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <set>.

Definition in file stl_multiset.h.

5.528 stl_numeric.h File Reference

Namespaces

- namespace std

Functions

- template< typename _InputIterator , typename _Tp >
  _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)
- template< typename _InputIterator , typename _Tp , typename _BinaryOperation >
  _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template< typename _InputIterator , typename _OutputIterator >
  _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template< typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template< typename _InputIterator1 , typename _InputIterator2 , typename _Tp >
  _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
- template< typename _InputIterator1 , typename _InputIterator2 , typename _Tp , typename _BinaryOperation1 , typename _BinaryOperation2 >
  _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)
- template< typename _ForwardIterator , typename _Tp >
  void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template< typename _InputIterator , typename _OutputIterator >
  _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template< typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
  _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)

5.528.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <numeric>.

Definition in file stl_numeric.h.

5.529 stl_pair.h File Reference

Classes

- struct std::pair< _T1, _T2 >

  Struct holding two objects of arbitrary type.
- struct std::piecewise_construct_t

  piecewise_construct_t
- class std::tuple< _Elements >

  Primary class template, tuple.

Namespaces

- namespace std

Functions

- template<class _T1 , class _T2 >
  constexpr pair< typename
  __decay_and_strip< _T1 >
  ::__type, typename
  __decay_and_strip< _T2 >
  ::__type > std::make_pair (_T1 &&__x, _T2 &&__y)
- template<class _T1 , class _T2 >
  constexpr bool std::operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<class _T1 , class _T2 >
  constexpr bool std::operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<class _T1 , class _T2 >
  constexpr bool std::operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<class _T1 , class _T2 >
  constexpr bool std::operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<class _T1 , class _T2 >
  constexpr bool std::operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<class _T1 , class _T2 >
  constexpr bool std::operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)
- template<class _T1 , class _T2 >
  void std::swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))

Variables

- constexpr piecewise_construct_t std::piecewise_construct

5.529.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

Definition in file stl_pair.h.

5.530 stl_queue.h File Reference

Classes

- class std::priority_queue< _Tp, _Sequence, _Compare >

  A standard container automatically sorting its contents.
- class std::queue< _Tp, _Sequence >

  A standard container giving FIFO behavior.

Namespaces

- namespace std

Functions

- template<typename _Tp , typename _Seq >
  bool std::operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  void **std::swap** (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp , typename _Sequence , typename _Compare >
  void **std::swap** (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))

5.530.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <queue>.

Definition in file stl_queue.h.

5.531 stl_raw_storage_iter.h File Reference

Classes

- class std::raw_storage_iterator< _OutputIterator, _Tp >

Namespaces

- namespace std

5.531.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file stl_raw_storage_iter.h.

5.532 stl_relops.h File Reference

Namespaces

- namespace std
- namespace std::rel_ops

Functions

- template<class _Tp >
  bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
  bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
  bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)
- template<class _Tp >
  bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)

5.532.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except std_utility.h. For more information, see the thread of about twenty messages starting with `http://gcc.gnu.org/ml/libstdc++/2001-01/msg00223.html`, or `http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#faq.ambiguous_overloads`

Short summary: the rel_ops operators should be avoided for the present.

Definition in file stl_relops.h.

5.533 stl_set.h File Reference

Classes

- class std::set< _Key, _Compare, _Alloc >

  A standard container made up of unique keys, which can be retrieved in logarithmic time.

Namespaces

- namespace std

Functions

- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Compare , typename _Alloc >
  void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y)

5.533.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <set>.

Definition in file stl_set.h.

5.534 stl_stack.h File Reference

Classes

- class std::stack< _Tp, _Sequence >

  A standard container giving FILO behavior.

Namespaces

- namespace std

Functions

- template<typename _Tp , typename _Seq >
  bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp , typename _Seq >
  void **std::swap** (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))

5.534.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stack>.

Definition in file stl_stack.h.

5.535 stl_tempbuf.h File Reference

Classes

- class std::_Temporary_buffer< _ForwardIterator, _Tp >

Namespaces

- namespace std

Functions

- template<typename _Pointer , typename _ForwardIterator >
  void **std::__uninitialized_construct_buf** (_Pointer __first, _Pointer __last, _ForwardIterator __seed)
- template<typename _Tp >
  pair< _Tp ∗, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len) noexcept
- template<typename _Tp >
  void std::return_temporary_buffer (_Tp ∗__p)

5.535.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file stl_tempbuf.h.

5.536 stl_tree.h File Reference

Namespaces

- namespace std

Enumerations

- enum **_Rb_tree_color** { **_S_red**, **_S_black** }

Functions

- unsigned int **std::_Rb_tree_black_count** (const _Rb_tree_node_base ∗__node, const _Rb_tree_node_base ∗__root) throw ()
- _Rb_tree_node_base ∗ **std::_Rb_tree_decrement** (_Rb_tree_node_base ∗__x) throw ()
- const _Rb_tree_node_base ∗ **std::_Rb_tree_decrement** (const _Rb_tree_node_base ∗__x) throw ()
- _Rb_tree_node_base ∗ **std::_Rb_tree_increment** (_Rb_tree_node_base ∗__x) throw ()
- const _Rb_tree_node_base ∗ **std::_Rb_tree_increment** (const _Rb_tree_node_base ∗__x) throw ()
- void **std::_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base ∗__x, _Rb_tree_node_base ∗__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base ∗ **std::_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base ∗const __z, _Rb_tree_node_base &__header) throw ()
- template<typename _Val >
  bool **std::operator!=** (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **std::operator!=** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **std::operator<** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **std::operator<=** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Val >
  bool **std::operator==** (const _Rb_tree_iterator< _Val > &__x, const _Rb_tree_const_iterator< _Val > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **std::operator==** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **std::operator>** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  bool **std::operator>=** (const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, const _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Key , typename _Val , typename _KeyOfValue , typename _Compare , typename _Alloc >
  void **std::swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)

5.536.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map> or <set>.

Definition in file stl_tree.h.

5.537 stl_uninitialized.h File Reference

Namespaces

- namespace std

Functions

- template<typename _InputIterator , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **std::__uninitialized_copy_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
- template<typename _InputIterator , typename _ForwardIterator , typename _Tp >
  _ForwardIterator **std::__uninitialized_copy_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, allocator< _Tp > &)
- template<typename _InputIterator1 , typename _InputIterator2 , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **std::__uninitialized_copy_move** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)
- template<typename _InputIterator , typename _Size , typename _ForwardIterator >
  _ForwardIterator **std::__uninitialized_copy_n** (_InputIterator __first, _Size __n, _ForwardIterator __result, input_iterator_tag)
- template<typename _RandomAccessIterator , typename _Size , typename _ForwardIterator >
  _ForwardIterator **std::__uninitialized_copy_n** (_RandomAccessIterator __first, _Size __n, _ForwardIterator __result, random_access_iterator_tag)
- template<typename _ForwardIterator >
  void **std::__uninitialized_default** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator , typename _Allocator >
  void **std::__uninitialized_default_a** (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Tp >
  void **std::__uninitialized_default_a** (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)
- template<typename _ForwardIterator , typename _Size >
  void **std::__uninitialized_default_n** (_ForwardIterator __first, _Size __n)
- template<typename _ForwardIterator , typename _Size , typename _Allocator >
  void **std::__uninitialized_default_n_a** (_ForwardIterator __first, _Size __n, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Size , typename _Tp >
  void **std::__uninitialized_default_n_a** (_ForwardIterator __first, _Size __n, allocator< _Tp > &)
- template<typename _ForwardIterator , typename _Tp , typename _Allocator >
  void **std::__uninitialized_fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, _Allocator &__alloc)
- template<typename _ForwardIterator , typename _Tp , typename _Tp2 >
  void **std::__uninitialized_fill_a** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x, allocator< _Tp2 > &)
- template<typename _ForwardIterator , typename _Tp , typename _InputIterator , typename _Allocator >
  _ForwardIterator **std::__uninitialized_fill_move** (_ForwardIterator __result, _ForwardIterator __mid, const _Tp &__x, _InputIterator __first, _InputIterator __last, _Allocator &__alloc)

- template< typename _ForwardIterator , typename _Size , typename _Tp , typename _Allocator >
  void **std::__uninitialized_fill_n_a** (_ForwardIterator __first, _Size __n, const _Tp &__x, _Allocator &__alloc)
- template< typename _ForwardIterator , typename _Size , typename _Tp , typename _Tp2 >
  void **std::__uninitialized_fill_n_a** (_ForwardIterator __first, _Size __n, const _Tp &__x, allocator< _Tp2 > &)
- template< typename _InputIterator , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **std::__uninitialized_move_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
- template< typename _InputIterator1 , typename _InputIterator2 , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **std::__uninitialized_move_copy** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _ForwardIterator __result, _Allocator &__alloc)
- template< typename _InputIterator , typename _ForwardIterator , typename _Tp , typename _Allocator >
  void **std::__uninitialized_move_fill** (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, const _Tp &__x, _Allocator &__alloc)
- template< typename _InputIterator , typename _ForwardIterator , typename _Allocator >
  _ForwardIterator **std::__uninitialized_move_if_noexcept_a** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _Allocator &__alloc)
- template< typename _InputIterator , typename _ForwardIterator >
  _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template< typename _InputIterator , typename _Size , typename _ForwardIterator >
  _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template< typename _ForwardIterator , typename _Tp >
  void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template< typename _ForwardIterator , typename _Size , typename _Tp >
  void std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)

5.537.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file stl_uninitialized.h.

5.538 stl_vector.h File Reference

Classes

- struct std::_Vector_base< _Tp, _Alloc >

  See bits/stl_deque.h's _Deque_base for an explanation.
- class std::vector< _Tp, _Alloc >

  A standard container which offers fixed time access to individual elements in any order.

Namespaces

- namespace std

Functions

- template< typename _Tp , typename _Alloc >
  bool std::operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template< typename _Tp , typename _Alloc >
  bool std::operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)

- template<typename _Tp , typename _Alloc >
  bool std::operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  bool std::operator>= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp , typename _Alloc >
  void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y)

5.538.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

Definition in file stl_vector.h.

5.539 stream_iterator.h File Reference

Classes

- class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >

  Provides input iterator semantics for streams.
- class std::ostream_iterator< _Tp, _CharT, _Traits >

  Provides output iterator semantics for streams.

Namespaces

- namespace std

Functions

- template<class _Tp , class _CharT , class _Traits , class _Dist >
  bool std::operator!= (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)
- template<typename _Tp , typename _CharT , typename _Traits , typename _Dist >
  bool std::operator== (const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__x, const istream_iterator< _Tp, _CharT, _Traits, _Dist > &__y)

5.539.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

Definition in file stream_iterator.h.

5.540 streambuf File Reference

Classes

- class std::basic_streambuf< _CharT, _Traits >

  The actual work of input and output (interface).

Namespaces

- namespace std

Macros

- #define **_GLIBXX_STREAMBUF**

Functions

- template<typename _CharT , typename _Traits >
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > ∗, basic_streambuf< _CharT, _Traits > ∗, bool &)
- template<>
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< char > ∗__sbin, basic_streambuf< char > ∗__sbout, bool &__ineof)
- template<>
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< wchar_t > ∗__sbin, basic_streambuf< wchar_t > ∗__sbout, bool &__ineof)

5.540.1 Detailed Description

This is a Standard C++ Library header.

Definition in file streambuf.

5.541 streambuf.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_STREAMBUF_TCC**

Functions

- template<typename _CharT , typename _Traits >
  streamsize **std::__copy_streambufs** (basic_streambuf< _CharT, _Traits > ∗__sbin, basic_streambuf< _CharT, _Traits > ∗__sbout)

- template<typename _CharT , typename _Traits >
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > ∗, basic_streambuf< _CharT, _Traits > ∗, bool &)

5.541.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <streambuf>.

Definition in file streambuf.tcc.

5.542 streambuf_iterator.h File Reference

Classes

- class std::istreambuf_iterator< _CharT, _Traits >
  Provides input iterator semantics for streambufs.
- class std::ostreambuf_iterator< _CharT, _Traits >
  Provides output iterator semantics for streambufs.

Namespaces

- namespace std

Functions

- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **std::__copy_move_a2** (_CharT ∗__first, _CharT ∗__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **std::__copy_move_a2** (const _CharT ∗__first, const _CharT ∗__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  _CharT ∗ >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT ∗__result)
- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  ostreambuf_iterator< _CharT >
  >::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)

- template<typename _CharT >
  __gnu_cxx::__enable_if
  < __is_char< _CharT >::__value,
  istreambuf_iterator< _CharT >
  >::__type **std::find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _CharT , typename _Traits >
  bool **std::operator!=** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _CharT , typename _Traits >
  bool **std::operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)

5.542.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

Definition in file streambuf_iterator.h.

5.543 string File Reference

Macros

- #define **_GLIBCXX_STRING**

5.543.1 Detailed Description

This is a Standard C++ Library header.

Definition in file string.

5.544 string File Reference

Classes

- class __gnu_debug::basic_string< _CharT, _Traits, _Allocator >

  Class std::basic_string with safety/checking/debug instrumentation.

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_STRING**

Typedefs

- typedef basic_string< char > **__gnu_debug::string**
- typedef basic_string< wchar_t > **__gnu_debug::wstring**

Functions

- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_debug::getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_debug::getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator!=** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **__gnu_debug::operator+** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **__gnu_debug::operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  basic_string< _CharT, _Traits,
  _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator<** (const _CharT ∗__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT ∗__rhs)

- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_ostream< _CharT,
  _Traits > & **__gnu_debug::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator<=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator==** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator>** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator>=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  bool **__gnu_debug::operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Allocator >
  std::basic_istream< _CharT,
  _Traits > & **__gnu_debug::operator>>** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT , typename _Traits , typename _Allocator >
  void **__gnu_debug::swap** (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &__rhs)

5.544.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/string.

5.545 string_conversions.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _TRet , typename _Ret = _TRet, typename _CharT , typename... _Base>
  _Ret **__gnu_cxx::__stoa** (_TRet(∗__convf)(const _CharT ∗, _CharT ∗∗, _Base...), const char ∗__name, const _CharT ∗__str, std::size_t ∗__idx, _Base...__base)
- template<typename _String , typename _CharT = typename _String::value_type>
  _String **__gnu_cxx::__to_xstring** (int(∗__convf)(_CharT ∗, std::size_t, const _CharT ∗, __builtin_va_list), std::size_t __n, const _CharT ∗__fmt,...)

5.545.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file string_conversions.h.

5.546 stringfwd.h File Reference

Classes

- struct std::allocator< _Tp >

  The standard allocator, as per [20.4].
- class std::basic_string< _CharT, _Traits, _Alloc >

  Managing sequences of characters and character-like objects.
- struct std::char_traits< _CharT >

  Basis for explicit traits specializations.

Namespaces

- namespace std

Typedefs

- typedef basic_string< char > std::string
- typedef basic_string< char16_t > std::u16string
- typedef basic_string< char32_t > std::u32string
- typedef basic_string< wchar_t > std::wstring

5.546.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

Definition in file stringfwd.h.

5.547 strstream File Reference

Namespaces

- namespace std

5.547.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <sstream>.

Definition in file strstream.

5.548 synth_access_traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >

  Synthetic element access traits.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC**
- #define **PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC**

5.548.1 Detailed Description

Contains an implementation class for a patricia tree.

Definition in file synth_access_traits.hpp.

5.549 system_error File Reference

Classes

- class std::error_category

  error_category
- struct std::error_code

  error_code
- struct std::error_condition

  error_condition
- struct std::hash< error_code >

  std::hash specialization for error_code.
- struct std::is_error_code_enum< _Tp >

  is_error_code_enum

- struct std::is_error_condition_enum< _Tp >

    is_error_condition_enum
- class std::system_error

    Thrown to indicate error code of underlying system.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_SYSTEM_ERROR**

Functions

- const error_category & **std::generic_category** () noexcept
- error_code **std::make_error_code** (errc __e) noexcept
- error_condition **std::make_error_condition** (errc __e) noexcept
- bool **std::operator!=** (const error_code &__lhs, const error_code &__rhs) noexcept
- bool **std::operator!=** (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool **std::operator!=** (const error_condition &__lhs, const error_code &__rhs) noexcept
- bool **std::operator!=** (const error_condition &__lhs, const error_condition &__rhs) noexcept
- bool **std::operator<** (const error_code &__lhs, const error_code &__rhs) noexcept
- bool **std::operator<** (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _CharT , typename _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- bool **std::operator==** (const error_code &__lhs, const error_code &__rhs) noexcept
- bool **std::operator==** (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool **std::operator==** (const error_condition &__lhs, const error_code &__rhs) noexcept
- bool **std::operator==** (const error_condition &__lhs, const error_condition &__rhs) noexcept
- const error_category & **std::system_category** () noexcept

Variables

- error_code **std::make_error_code** (errc) noexcept
- error_condition **std::make_error_condition** (errc) noexcept

5.549.1 Detailed Description

This is a Standard C++ Library header.

Definition in file system_error.

5.550 tag_and_trait.hpp File Reference

Classes

- struct __gnu_pbds::associative_tag

  Basic associative-container.
- struct __gnu_pbds::basic_branch_tag

  Basic branch structure.
- struct __gnu_pbds::basic_hash_tag

  Basic hash structure.
- struct __gnu_pbds::basic_invalidation_guarantee
- struct __gnu_pbds::binary_heap_tag

  Binary-heap (array-based).
- struct __gnu_pbds::binomial_heap_tag

  Binomial-heap.
- struct __gnu_pbds::cc_hash_tag

  Collision-chaining hash.
- struct __gnu_pbds::container_tag

  Base data structure tag.
- struct __gnu_pbds::container_traits< Cntnr >

  Container traits.
- struct __gnu_pbds::container_traits_base< binary_heap_tag >

  Specialization, binary heap.
- struct __gnu_pbds::container_traits_base< binomial_heap_tag >

  Specialization, binomial heap.
- struct __gnu_pbds::container_traits_base< cc_hash_tag >

  Specialization, cc hash.
- struct __gnu_pbds::container_traits_base< gp_hash_tag >

  Specialization, gp hash.
- struct __gnu_pbds::container_traits_base< list_update_tag >

  Specialization, list update.
- struct __gnu_pbds::container_traits_base< ov_tree_tag >

  Specialization, ov tree.
- struct __gnu_pbds::container_traits_base< pairing_heap_tag >

  Specialization, pairing heap.
- struct __gnu_pbds::container_traits_base< pat_trie_tag >

  Specialization, pat trie.
- struct __gnu_pbds::container_traits_base< rb_tree_tag >

  Specialization, rb tree.
- struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >

  Specialization, rc binomial heap.
- struct __gnu_pbds::container_traits_base< splay_tree_tag >

  Specialization, splay tree.
- struct __gnu_pbds::container_traits_base< thin_heap_tag >

  Specialization, thin heap.
- struct __gnu_pbds::gp_hash_tag

  General-probing hash.
- struct __gnu_pbds::list_update_tag

List-update.

- struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >

  A null node updator, indicating that no node updates are required.
- struct __gnu_pbds::null_type

  Represents no type, or absence of type, for template tricks.
- struct __gnu_pbds::ov_tree_tag

  Ordered-vector tree.
- struct __gnu_pbds::pairing_heap_tag

  Pairing-heap.
- struct __gnu_pbds::pat_trie_tag

  PATRICIA trie.
- struct __gnu_pbds::point_invalidation_guarantee
- struct __gnu_pbds::priority_queue_tag

  Basic priority-queue.
- struct __gnu_pbds::range_invalidation_guarantee
- struct __gnu_pbds::rb_tree_tag

  Red-black tree.
- struct __gnu_pbds::rc_binomial_heap_tag

  Redundant-counter binomial-heap.
- struct __gnu_pbds::sequence_tag

  Basic sequence.
- struct __gnu_pbds::splay_tree_tag

  Splay tree.
- struct __gnu_pbds::string_tag

  Basic string container, inclusive of strings, ropes, etc.
- struct __gnu_pbds::thin_heap_tag

  Thin heap.
- struct __gnu_pbds::tree_tag

  Basic tree structure.
- struct __gnu_pbds::trie_tag

  Basic trie structure.
- struct __gnu_pbds::trivial_iterator_tag

  A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.
- struct container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >

  Dispatch mechanism, primary template for associative types.
- struct container_traits_base< _Tag >

  Primary template, container traits base.

Namespaces

- namespace __gnu_pbds

Typedefs

- typedef void __gnu_pbds::trivial_iterator_difference_type

5.550.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

Definition in file tag_and_trait.hpp.

5.551 tags.h File Reference

Classes

- struct __gnu_parallel::balanced_quicksort_tag

  Forces parallel sorting using balanced quicksort at compile time.
- struct __gnu_parallel::balanced_tag

  Recommends parallel execution using dynamic load-balancing at compile time.
- struct __gnu_parallel::constant_size_blocks_tag

  Selects the constant block size variant for std::find().
- struct __gnu_parallel::default_parallel_tag

  Recommends parallel execution using the default parallel algorithm.
- struct __gnu_parallel::equal_split_tag

  Selects the equal splitting variant for std::find().
- struct __gnu_parallel::exact_tag

  Forces parallel merging with exact splitting, at compile time.
- struct __gnu_parallel::find_tag

  Base class for for std::find() variants.
- struct __gnu_parallel::growing_blocks_tag

  Selects the growing block size variant for std::find().
- struct __gnu_parallel::multiway_mergesort_exact_tag

  Forces parallel sorting using multiway mergesort with exact splitting at compile time.
- struct __gnu_parallel::multiway_mergesort_sampling_tag

  Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.
- struct __gnu_parallel::multiway_mergesort_tag

  Forces parallel sorting using multiway mergesort at compile time.
- struct __gnu_parallel::omp_loop_static_tag

  Recommends parallel execution using OpenMP static load-balancing at compile time.
- struct __gnu_parallel::omp_loop_tag

  Recommends parallel execution using OpenMP dynamic load-balancing at compile time.
- struct __gnu_parallel::parallel_tag

  Recommends parallel execution at compile time, optionally using a user-specified number of threads.
- struct __gnu_parallel::quicksort_tag

  Forces parallel sorting using unbalanced quicksort at compile time.
- struct __gnu_parallel::sampling_tag

  Forces parallel merging with exact splitting, at compile time.
- struct __gnu_parallel::sequential_tag

  Forces sequential execution at compile time.
- struct __gnu_parallel::unbalanced_tag

  Recommends parallel execution using static load-balancing at compile time.

Namespaces

- namespace __gnu_parallel

5.551.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file tags.h.

5.552 tgmath.h File Reference

Macros

- #define **_GLIBCXX_TGMATH_H**

5.552.1 Detailed Description

This is a Standard C++ Library header.

Definition in file tgmath.h.

5.553 thin_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_AUX_NULL**(X)
- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_BASE_T_P**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

Enumerations

- enum { **num_distinct_rank_bounds** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_rank_bounds** [num_distinct_rank_bounds]

5.553.1 Detailed Description

Contains an implementation class for a thin heap.

Definition in file thin_heap_.hpp.

5.554 thread File Reference

Classes

- struct std::hash< thread::id >

  std::hash specialization for thread::id.
- class std::thread

  thread
- class std::thread::id

  thread::id

Namespaces

- namespace std
- namespace std::this_thread

Macros

- #define **_GLIBCXX_THREAD**

Functions

- void **std::this_thread::__sleep_for** (chrono::seconds, chrono::nanoseconds)
- thread::id std::this_thread::get_id () noexcept
- bool **std::operator!=** (thread::id __x, thread::id __y) noexcept
- template<class _CharT , class _Traits >
  basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__out, thread::id __id)
- bool **std::operator<=** (thread::id __x, thread::id __y) noexcept
- bool **std::operator>** (thread::id __x, thread::id __y) noexcept
- bool **std::operator>=** (thread::id __x, thread::id __y) noexcept
- template<typename _Rep , typename _Period >
  void std::this_thread::sleep_for (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock , typename _Duration >
  void std::this_thread::sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)
- void **std::swap** (thread &__x, thread &__y) noexcept
- void std::this_thread::yield () noexcept

5.554.1 Detailed Description

This is a Standard C++ Library header.

Definition in file thread.

5.555 throw_allocator.h File Reference

Classes

- struct __gnu_cxx::annotate_base

  Base class for checking address and label information about allocations. Create a std::map between the allocated address (void∗) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.
- struct __gnu_cxx::condition_base

  Base struct for condition policy.
- struct __gnu_cxx::forced_error

  Thown by exception safety machinery.
- struct __gnu_cxx::limit_condition

  Base class for incremental control and throw.
- struct __gnu_cxx::limit_condition::always_adjustor

  Always enter the condition.
- struct __gnu_cxx::limit_condition::limit_adjustor

  Enter the nth condition.
- struct __gnu_cxx::limit_condition::never_adjustor

  Never enter the condition.
- struct __gnu_cxx::random_condition

  Base class for random probability control and throw.
- struct __gnu_cxx::random_condition::always_adjustor

  Always enter the condition.
- struct __gnu_cxx::random_condition::group_adjustor

  Group condition.
- struct __gnu_cxx::random_condition::never_adjustor

  Never enter the condition.
- class __gnu_cxx::throw_allocator_base< _Tp, _Cond >

  Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templatized code.
  Note: Deallocate not allowed to throw.
- struct __gnu_cxx::throw_allocator_limit< _Tp >

  Allocator throwing via limit condition.
- struct __gnu_cxx::throw_allocator_random< _Tp >

  Allocator throwing via random condition.
- struct __gnu_cxx::throw_value_base< _Cond >

  Class with exception generation control. Intended to be used as a value_type in templatized code.
- struct __gnu_cxx::throw_value_limit

  Type throwing via limit condition.
- struct __gnu_cxx::throw_value_random

  Type throwing via random condition.
- struct std::hash< __gnu_cxx::throw_value_limit >

  Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.
- struct std::hash< __gnu_cxx::throw_value_random >

  Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- void **__gnu_cxx::__throw_forced_error** ()
- template<typename _Tp , typename _Cond >
  bool **__gnu_cxx::operator!=** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
  throw_value_base< _Cond > **__gnu_cxx::operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
  throw_value_base< _Cond > **__gnu_cxx::operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
  throw_value_base< _Cond > **__gnu_cxx::operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond >
  bool **__gnu_cxx::operator<** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- std::ostream & **__gnu_cxx::operator<<** (std::ostream &os, const annotate_base &__b)
- template<typename _Cond >
  bool **__gnu_cxx::operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp , typename _Cond >
  bool **__gnu_cxx::operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond >
  void **__gnu_cxx::swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)

5.555.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (throw_value, throw_allocator) intended to be used as value and allocator types while testing exception safety in templatized containers and algorithms. The allocator has additional log and debug features. The exception generated is of type forced_exception_error.

Definition in file throw_allocator.h.

5.556 time_members.h File Reference

Namespaces

- namespace std

5.556.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

Definition in file time_members.h.

5.557 trace_fn_imps.hpp File Reference

5.557.1 Detailed Description

Contains an implementation class for a binary_heap.

Definition in file binary_heap_/trace_fn_imps.hpp.

5.558 trace_fn_imps.hpp File Reference

5.558.1 Detailed Description

Contains implementations of cc_ht_map_'s trace-mode functions.

Definition in file cc_hash_table_map_/trace_fn_imps.hpp.

5.559 trace_fn_imps.hpp File Reference

5.559.1 Detailed Description

Contains implementations of gp_ht_map_'s trace-mode functions.

Definition in file gp_hash_table_map_/trace_fn_imps.hpp.

5.560 trace_fn_imps.hpp File Reference

5.560.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file left_child_next_sibling_heap_/trace_fn_imps.hpp.

5.561 trace_fn_imps.hpp File Reference

5.561.1 Detailed Description

Contains implementations of lu_map_.

Definition in file list_update_map_/trace_fn_imps.hpp.

5.562 trace_fn_imps.hpp File Reference

5.562.1 Detailed Description

Contains an implementation class for pat_trie_.

Definition in file pat_trie_/trace_fn_imps.hpp.

5.563 trace_fn_imps.hpp File Reference

5.563.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

Definition in file rc_binomial_heap_/trace_fn_imps.hpp.

5.564 trace_fn_imps.hpp File Reference

5.564.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

Definition in file thin_heap_/trace_fn_imps.hpp.

5.565 traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >

  Binary search tree traits, primary template.
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

  Specialization.

Namespaces

- namespace __gnu_pbds

5.565.1 Detailed Description

Contains an implementation for bin_search_tree_.

Definition in file bin_search_tree_/traits.hpp.

5.566 traits.hpp File Reference

Classes

- struct tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >

  Tree traits class, primary template.
- struct trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >

  Trie traits class, primary template.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_DEBUG_VERIFY**(_Cond)

5.566.1 Detailed Description

Contains an implementation class for tree-like classes.

Definition in file branch_policy/traits.hpp.

5.567 traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

  Tree traits.
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

  Specialization.

Namespaces

- namespace __gnu_pbds

5.567.1 Detailed Description

Contains an implementation class for ov_tree_.

Definition in file ov_tree_map_/traits.hpp.

5.568 traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

  Specialization.

Namespaces

- namespace __gnu_pbds

5.568.1 Detailed Description

Contains an implementation class for pat_trie_.

Definition in file pat_trie_/traits.hpp.

5.569 traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

  Specialization.

Namespaces

- namespace __gnu_pbds

5.569.1 Detailed Description

Contains an implementation for rb_tree_.

Definition in file rb_tree_map_/traits.hpp.

5.570 traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

  Specialization.
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

  Specialization.

Namespaces

- namespace __gnu_pbds

5.570.1 Detailed Description

Contains an implementation for splay_tree_.

Definition in file splay_tree_/traits.hpp.

5.571 tree_policy.hpp File Reference

Classes

- class __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >

  Functor updating ranks of entrees.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BRANCH_POLICY_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.571.1 Detailed Description

Contains tree-related policies.

Definition in file tree_policy.hpp.

5.572 tree_trace_base.hpp File Reference

5.572.1 Detailed Description

Contains tree-related policies.

Definition in file tree_trace_base.hpp.

5.573 trie_policy.hpp File Reference

Classes

- class __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

  Functor updating ranks of entrees.
- class __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

  A node updator that allows tries to be searched for the range of values that match a certain prefix.
- struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_TRIE_POLICY_BASE**

5.573.1 Detailed Description

Contains trie-related policies.

Definition in file trie_policy.hpp.

5.574 trie_policy_base.hpp File Reference

Classes

- class __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >

  Base class for trie policies.

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

5.574.1 Detailed Description

Contains an implementation of trie_policy_base.

Definition in file trie_policy_base.hpp.

5.575 trie_string_access_traits_imp.hpp File Reference

5.575.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

Definition in file trie_string_access_traits_imp.hpp.

5.576 tuple File Reference

Classes

- struct _Tuple_impl< _Idx, _Elements >
- struct std::_Tuple_impl< _Idx >
- struct std::_Tuple_impl< _Idx, _Head, _Tail...>
- class std::tuple< _Elements >

  Primary class template, tuple.
- class std::tuple< _T1, _T2 >

  Partial specialization, 2-element tuple. Includes construction and assignment from a pair.
- struct std::tuple_element< 0, tuple< _Head, _Tail...> >
- struct std::tuple_element< __i, tuple< _Head, _Tail...> >

- struct std::tuple_size< tuple< _Elements...> >

  class tuple_size
- struct std::uses_allocator< tuple< _Types...>, _Alloc >

  Partial specialization for tuples.
- struct tuple_element< __i, _Tp >

  Gives the type of the ith element of a given tuple type.
- struct tuple_size< _Tp >

  Finds the size of a given tuple type.

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_TUPLE**

Typedefs

- template<typename _Tp >
  using **std::__empty_not_final** = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp >>::type

Functions

- template<std::size_t __i, typename _Head , typename... _Tail>
  constexpr __add_ref< _Head >::type **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail...> &__t) noexcept
- template<std::size_t __i, typename _Head , typename... _Tail>
  constexpr __add_c_ref< _Head >
  ::type **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail...> &__t) noexcept
- template<typename... _Elements>
  tuple< _Elements &&...> **std::forward_as_tuple** (_Elements &&...__args) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_ref< typename
  tuple_element< __i, tuple
  < _Elements...> >::type >
  ::type **std::get** (tuple< _Elements...> &__t) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_c_ref
  < typename tuple_element< __i,
  tuple< _Elements...> >::type >
  ::type **std::get** (const tuple< _Elements...> &__t) noexcept
- template<std::size_t __i, typename... _Elements>
  constexpr __add_r_ref
  < typename tuple_element< __i,
  tuple< _Elements...> >::type >
  ::type **std::get** (tuple< _Elements...> &&__t) noexcept
- template<typename... _Elements>
  constexpr tuple< typename
  __decay_and_strip< _Elements >
  ::__type...> **std::make_tuple** (_Elements &&...__args)

- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator!=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**< (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**<**=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator==** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**> (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename... _TElements, typename... _UElements>
  constexpr bool **std::operator**>**=** (const tuple< _TElements...> &__t, const tuple< _UElements...> &__u)
- template<typename... _Elements>
  void std::swap (tuple< _Elements...> &__x, tuple< _Elements...> &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  tuple< _Elements &...> std::tie (_Elements &...__args) noexcept
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto std::tuple_cat (_Tpls &&...__tpls) -> typename __tuple_cat_result< _Tpls...>::__type

Variables

- const _Swallow_assign **std::ignore**

5.576.1 Detailed Description

This is a Standard C++ Library header.

Definition in file tuple.

5.577 type_traits File Reference

Classes

- struct common_type< _Tp >

  common_type
- class result_of< _Signature >

  result_of
- struct std::add_const< _Tp >

  add_const
- struct std::add_cv< _Tp >

  add_cv
- struct std::add_lvalue_reference< _Tp >

  add_lvalue_reference
- struct std::add_pointer< _Tp >

  add_pointer
- struct std::add_rvalue_reference< _Tp >

  add_rvalue_reference
- struct std::add_volatile< _Tp >

  add_volatile

- struct std::aligned_storage< _Len, _Align >

  Alignment type.
- struct std::alignment_of< _Tp >

  alignment_of
- struct std::conditional< _Cond, _Iftrue, _Iffalse >

  Define a member typedef `type` to one of two argument types.
- class std::decay< _Tp >

  decay
- struct std::enable_if< bool, _Tp >

  Define a member typedef `type` only if a boolean constant is true.
- struct std::extent< typename, _Uint >

  extent
- struct std::has_trivial_copy_assign< _Tp >

  has_trivial_copy_assign (temporary legacy)
- struct std::has_trivial_copy_constructor< _Tp >

  has_trivial_copy_constructor (temporary legacy)
- struct std::has_trivial_default_constructor< _Tp >

  has_trivial_default_constructor (temporary legacy)
- struct std::has_virtual_destructor< _Tp >

  has_virtual_destructor
- struct std::integral_constant< _Tp, __v >

  integral_constant
- struct std::is_abstract< _Tp >

  is_abstract
- struct std::is_arithmetic< _Tp >

  is_arithmetic
- struct std::is_array< typename >

  is_array
- struct std::is_assignable< _Tp, _Up >

  is_assignable
- struct std::is_base_of< _Base, _Derived >

  is_base_of
- struct std::is_class< _Tp >

  is_class
- struct std::is_compound< _Tp >

  is_compound
- struct std::is_const< typename >

  is_const
- struct std::is_constructible< _Tp, _Args >

  is_constructible
- struct std::is_convertible< _From, _To >

  is_convertible
- struct std::is_copy_assignable< _Tp >

  is_copy_assignable
- struct std::is_copy_constructible< _Tp >

  is_copy_constructible
- struct std::is_default_constructible< _Tp >

is_default_constructible

- struct std::is_destructible< _Tp >

  is_destructible
- struct std::is_empty< _Tp >

  is_empty
- struct std::is_enum< _Tp >

  is_enum
- struct std::is_floating_point< _Tp >

  is_floating_point
- struct std::is_function< typename >

  is_function
- struct std::is_fundamental< _Tp >

  is_fundamental
- struct std::is_integral< _Tp >

  is_integral
- struct std::is_literal_type< _Tp >

  is_literal_type
- struct std::is_lvalue_reference< typename >

  is_lvalue_reference
- struct std::is_member_function_pointer< _Tp >

  is_member_function_pointer
- struct std::is_member_object_pointer< _Tp >

  is_member_object_pointer
- struct std::is_member_pointer< _Tp >

  is_member_pointer
- struct std::is_move_assignable< _Tp >

  is_move_assignable
- struct std::is_move_constructible< _Tp >

  is_move_constructible
- struct std::is_nothrow_assignable< _Tp, _Up >

  is_nothrow_assignable
- struct std::is_nothrow_constructible< _Tp, _Args >

  is_nothrow_constructible
- struct std::is_nothrow_copy_assignable< _Tp >

  is_nothrow_copy_assignable
- struct std::is_nothrow_copy_constructible< _Tp >

  is_nothrow_copy_constructible
- struct std::is_nothrow_default_constructible< _Tp >

  is_nothrow_default_constructible
- struct std::is_nothrow_destructible< _Tp >

  is_nothrow_destructible
- struct std::is_nothrow_move_assignable< _Tp >

  is_nothrow_move_assignable
- struct std::is_nothrow_move_constructible< _Tp >

  is_nothrow_move_constructible
- struct std::is_object< _Tp >

  is_object

- struct std::is_pod< _Tp >

  is_pod
- struct std::is_pointer< _Tp >

  is_pointer
- struct std::is_polymorphic< _Tp >

  is_polymorphic
- struct std::is_reference< _Tp >

  is_reference
- struct std::is_rvalue_reference< typename >

  is_rvalue_reference
- struct std::is_same< typename, typename >

  is_same
- struct std::is_scalar< _Tp >

  is_scalar
- struct std::is_signed< _Tp >

  is_signed
- struct std::is_standard_layout< _Tp >

  is_standard_layout
- struct std::is_trivial< _Tp >

  is_trivial
- struct std::is_trivially_destructible< _Tp >

  is_trivially_constructible (still unimplemented)
- struct std::is_union< _Tp >

  is_union
- struct std::is_unsigned< _Tp >

  is_unsigned
- struct std::is_void< _Tp >

  is_void
- struct std::is_volatile< typename >

  is_volatile
- struct std::make_signed< _Tp >

  make_signed
- struct std::make_unsigned< _Tp >

  make_unsigned
- struct std::rank< typename >

  rank
- class std::reference_wrapper< _Tp >

  Primary class template for reference_wrapper.
- struct std::remove_all_extents< _Tp >

  remove_all_extents
- struct std::remove_const< _Tp >

  remove_const
- struct std::remove_cv< _Tp >

  remove_cv
- struct std::remove_extent< _Tp >

  remove_extent
- struct std::remove_pointer< _Tp >

remove_pointer
- struct std::remove_reference< _Tp >

  remove_reference
- struct std::remove_volatile< _Tp >

  remove_volatile
- struct std::underlying_type< _Tp >

  The underlying type of an enum.

Namespaces

- namespace std

Macros

- #define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)
- #define **_GLIBCXX_TYPE_TRAITS**

Typedefs

- template<typename... _Cond>
  using **std::_Require** = typename enable_if< __and_< _Cond...>::value >::type
- typedef integral_constant
  < bool, false > std::false_type
- typedef integral_constant
  < bool, true > std::true_type

Functions

- template<typename _Tp >
  add_rvalue_reference< _Tp >::type std::declval () noexcept

5.577.1 Detailed Description

This is a Standard C++ Library header.

Definition in file type_traits.

5.577.2 Macro Definition Documentation

5.577.2.1 #define _GLIBCXX_HAS_NESTED_TYPE(*_NTYPE*)

Use SFINAE to determine if the type _Tp has a publicly-accessible member type _NTYPE.

Definition at line 2044 of file type_traits.

5.578 type_traits File Reference

Classes

- struct __reflection_typelist< _Elements >
- struct std::tr2::__reflection_typelist< _First, _Rest...>

  Partial specialization.
- struct std::tr2::__reflection_typelist<>

  Specialization for an empty typelist.
- struct std::tr2::bases< _Tp >

  Sequence abstraction metafunctions for manipulating a typelist.
- struct std::tr2::direct_bases< _Tp >

  Enumerate all the direct base classes of a class. Form of a typelist.

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_TYPE_TRAITS**

5.578.1 Detailed Description

This is a TR2 C++ Library header.

Definition in file tr2/type_traits.

5.579 type_traits.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Type >
  bool **__gnu_cxx::__is_null_pointer** (_Type ∗__ptr)
- template<typename _Type >
  bool **__gnu_cxx::__is_null_pointer** (_Type)

5.579.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file type_traits.h.

5.580 type_utils.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_STATIC_ASSERT**(UNIQUE, E)

Typedefs

- typedef
  std::tr1::integral_constant
  < int, 0 > **__gnu_pbds::detail::false_type**
- typedef
  std::tr1::integral_constant
  < int, 1 > **__gnu_pbds::detail::true_type**

5.580.1 Detailed Description

Contains utilities for handnling types. All of these classes are based on Modern C++ by Andrei Alxandrescu.

Definition in file type_utils.hpp.

5.581 typeindex File Reference

Classes

- struct std::hash< type_index >

  std::hash specialization for type_index.
- struct std::type_index

  Class type_index
  The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_TYPEINDEX**

5.581.1 Detailed Description

This is a Standard C++ Library header.

Definition in file typeindex.

5.582 typeinfo File Reference

Classes

- class std::bad_cast

  Thrown during incorrect typecasting.
  If you attempt an invalid `dynamic_cast` *expression, an instance of this class (or something derived from this class) is thrown.*
- class std::bad_typeid

  Thrown when a NULL pointer in a `typeid` *expression is used.*
- class std::type_info

  Part of RTTI.

Namespaces

- namespace std

Macros

- #define **__GXX_MERGED_TYPEINFO_NAMES**
- #define **__GXX_TYPEINFO_EQUALITY_INLINE**
- #define **_TYPEINFO**

5.582.1 Detailed Description

This is a Standard C++ Library header.

Definition in file typeinfo.

5.583 typelist.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::typelist

Macros

- #define **_GLIBCXX_TYPELIST_CHAIN1**(X0)
- #define **_GLIBCXX_TYPELIST_CHAIN10**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)
- #define **_GLIBCXX_TYPELIST_CHAIN11**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)
- #define **_GLIBCXX_TYPELIST_CHAIN12**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)
- #define **_GLIBCXX_TYPELIST_CHAIN13**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)
- #define **_GLIBCXX_TYPELIST_CHAIN14**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)
- #define **_GLIBCXX_TYPELIST_CHAIN15**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)
- #define **_GLIBCXX_TYPELIST_CHAIN16**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)
- #define **_GLIBCXX_TYPELIST_CHAIN17**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)

- #define **_GLIBCXX_TYPELIST_CHAIN18**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)
- #define **_GLIBCXX_TYPELIST_CHAIN19**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)
- #define **_GLIBCXX_TYPELIST_CHAIN2**(X0, X1)
- #define **_GLIBCXX_TYPELIST_CHAIN20**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)
- #define **_GLIBCXX_TYPELIST_CHAIN3**(X0, X1, X2)
- #define **_GLIBCXX_TYPELIST_CHAIN4**(X0, X1, X2, X3)
- #define **_GLIBCXX_TYPELIST_CHAIN5**(X0, X1, X2, X3, X4)
- #define **_GLIBCXX_TYPELIST_CHAIN6**(X0, X1, X2, X3, X4, X5)
- #define **_GLIBCXX_TYPELIST_CHAIN7**(X0, X1, X2, X3, X4, X5, X6)
- #define **_GLIBCXX_TYPELIST_CHAIN8**(X0, X1, X2, X3, X4, X5, X6, X7)
- #define **_GLIBCXX_TYPELIST_CHAIN9**(X0, X1, X2, X3, X4, X5, X6, X7, X8)

Functions

- template<typename Fn , typename Typelist >
void **__gnu_cxx::typelist::apply** (Fn &, Typelist)
- template<typename Gn , typename Typelist >
void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)
- template<typename Gn , typename TypelistT , typename TypelistV >
void **__gnu_cxx::typelist::apply_generator** (Gn &, TypelistT, TypelistV)
- template<typename Fn , typename Typelist >
void **__gnu_cxx::typelist::apply_generator** (Fn &fn, Typelist)
- template<typename Fn , typename TypelistT , typename TypelistV >
void **__gnu_cxx::typelist::apply_generator** (Fn &fn, TypelistT, TypelistV)

5.583.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains typelist_chain definitions. Typelists are an idea by Andrei Alexandrescu.

Definition in file typelist.h.

5.584 types.h File Reference

Namespaces

- namespace __gnu_parallel

Typedefs

- typedef int64_t __gnu_parallel::_CASable
- typedef uint64_t __gnu_parallel::_SequenceIndex
- typedef uint16_t __gnu_parallel::_ThreadIndex

Enumerations

- enum __gnu_parallel::_AlgorithmStrategy { **heuristic**, **force_sequential**, **force_parallel** }
- enum __gnu_parallel::_FindAlgorithm { **GROWING_BLOCKS**, **CONSTANT_SIZE_BLOCKS**, **EQUAL_SPLIT** }
- enum __gnu_parallel::_MultiwayMergeAlgorithm { **LOSER_TREE** }
- enum __gnu_parallel::_Parallelism {
  __gnu_parallel::sequential, __gnu_parallel::parallel_unbalanced, __gnu_parallel::parallel_balanced, __gnu_parallel::parallel_omp_loop,
  __gnu_parallel::parallel_omp_loop_static, __gnu_parallel::parallel_taskqueue }
- enum __gnu_parallel::_PartialSumAlgorithm { **RECURSIVE**, **LINEAR** }
- enum __gnu_parallel::_SortAlgorithm { **MWMS**, **QS**, **QS_BALANCED** }
- enum __gnu_parallel::_SplittingAlgorithm { **SAMPLING**, **EXACT** }

Variables

- static const int __gnu_parallel::_CASable_bits
- static const _CASable __gnu_parallel::_CASable_mask

5.584.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

Definition in file types.h.

5.585 types_traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >

  Primary template.
- struct __gnu_pbds::detail::no_throw_copies< Key, null_type >

  Specialization.
- struct __gnu_pbds::detail::stored_data< _Tv, _Th >

  Primary template for representation of stored data. Two types of data can be stored: value and hash.
- struct __gnu_pbds::detail::stored_data< _Tv, null_type >

  Specialization for representation of stored data of just value type.
- struct __gnu_pbds::detail::stored_hash< _Th >

  Stored hash.
- struct __gnu_pbds::detail::stored_value< _Tv >

  Stored value.
- struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, false >
- struct __gnu_pbds::detail::type_base< Key, Mapped, _Alloc, true >
- struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, false >
- struct __gnu_pbds::detail::type_base< Key, null_type, _Alloc, true >
- struct __gnu_pbds::detail::type_dispatch< Key, Mapped, _Alloc, Store_Hash >

  Type base dispatch.
- struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

  Traits for abstract types.
- struct type_base< Key, Mapped, _Alloc, Store_Hash >

  Primary template.

Namespaces

- namespace __gnu_pbds

5.585.1 Detailed Description

Contains a traits class of types used by containers.

Definition in file types_traits.hpp.

5.586 unique_copy.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter , class _OutputIterator , class _BinaryPredicate >
  _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _IIter , class _OutputIterator >
  _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result)

5.586.1 Detailed Description

Parallel implementations of std::unique_copy(). This file is a GNU parallel extension to the Standard C++ Library.

Definition in file unique_copy.h.

5.587 unique_ptr.h File Reference

Classes

- struct std::default_delete< _Tp >

  Primary template, default_delete.
- struct std::default_delete< _Tp[]>

  Specialization, default_delete.
- struct std::hash< unique_ptr< _Tp, _Dp > >

  std::hash specialization for unique_ptr.
- class std::unique_ptr< _Tp, _Dp >

  20.7.1.2 unique_ptr for single objects.
- class std::unique_ptr< _Tp[], _Dp >

  20.7.1.3 unique_ptr for array objects with a runtime length

Namespaces

- namespace std

Functions

- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator!=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator!=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp , typename _Dp >
  bool **std::operator!=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator<** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator<** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator<** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator<=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator<=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator<=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator==** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator==** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp , typename _Dp >
  bool **std::operator==** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator>** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
  bool **std::operator>=** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp , typename _Dp >
  bool **std::operator>=** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp , typename _Dp >
  bool **std::operator>=** (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp , typename _Dp >
  void **std::swap** (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept

5.587.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

Definition in file unique_ptr.h.

5.588 unordered_map File Reference

Macros

- #define **_GLIBCXX_UNORDERED_MAP**

5.588.1 Detailed Description

This is a Standard C++ Library header.

Definition in file unordered_map.

5.589 unordered_map File Reference

Classes

- class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

  Class std::unordered_map with safety/checking/debug instrumentation.
- class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

  Class std::unordered_multimap with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_UNORDERED_MAP**

Functions

- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator!=** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator!=** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__debug::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__debug::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

5.589.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/unordered_map.

5.590 unordered_map File Reference

Classes

- class std::__profile::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

  Class std::unordered_map wrapper with performance instrumentation.
- class std::__profile::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

  Class std::unordered_multimap wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Macros

- #define **_GLIBCXX_BASE**
- #define **_GLIBCXX_BASE**
- #define **_GLIBCXX_PROFILE_UNORDERED_MAP**
- #define **_GLIBCXX_STD_BASE**
- #define **_GLIBCXX_STD_BASE**

Functions

- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator!=** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator!=** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__profile::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Tp , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__profile::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

5.590.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/unordered_map.

5.591 unordered_map.h File Reference

Classes

- class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

  A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.
- class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

  A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Namespaces

- namespace std

Typedefs

- template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
  using **std::__umap_hashtable** = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using std::__umap_traits = __detail::_Hashtable_traits< _Cache, false, true >
- template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
  using **std::__ummap_hashtable** = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using std::__ummap_traits = __detail::_Hashtable_traits< _Cache, false, false >

Functions

- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **std::operator!=** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **std::operator!=** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **std::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  bool **std::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  void **std::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >
  void **std::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)

5.591.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map>.

Definition in file unordered_map.h.

5.592 unordered_set File Reference

Macros

- #define **_GLIBCXX_UNORDERED_SET**

5.592.1 Detailed Description

This is a Standard C++ Library header.

Definition in file unordered_set.

5.593 unordered_set File Reference

Classes

- class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

  Class std::unordered_multiset with safety/checking/debug instrumentation.
- class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >

  Class std::unordered_set with safety/checking/debug instrumentation.

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_UNORDERED_SET**

Functions

- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator!=** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator!=** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)

- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__debug::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__debug::swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__debug::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)

5.593.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/unordered_set.

5.594 unordered_set File Reference

Classes

- class std::__profile::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

  Unordered_multiset wrapper with performance instrumentation.
- class std::__profile::unordered_set< _Key, _Hash, _Pred, _Alloc >

  Unordered_set wrapper with performance instrumentation.

Namespaces

- namespace std
- namespace std::__profile

Macros

- #define **_GLIBCXX_BASE**
- #define **_GLIBCXX_BASE**
- #define **_GLIBCXX_PROFILE_UNORDERED_SET**
- #define **_GLIBCXX_STD_BASE**
- #define **_GLIBCXX_STD_BASE**

Functions

- template<typename _Key , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator!=** (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator!=** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator==** (const unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)

- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  bool **std::__profile::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__profile::swap** (unordered_set< _Key, _Hash, _Pred, _Alloc > &__x, unordered_set< _Key, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value , typename _Hash , typename _Pred , typename _Alloc >
  void **std::__profile::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)

5.594.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/unordered_set.

5.595 unordered_set.h File Reference

Classes

- class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

  A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.
- class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

  A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Namespaces

- namespace std

Typedefs

- template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
  using **std::__umset_hashtable** = _Hashtable< _Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using std::__umset_traits = __detail::_Hashtable_traits< _Cache, true, false >
- template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
  using **std::__uset_hashtable** = _Hashtable< _Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >
- template<bool _Cache>
  using std::__uset_traits = __detail::_Hashtable_traits< _Cache, true, true >

Functions

- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **std::operator!=** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)

- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **std::operator!=** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **std::operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  bool **std::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  void **std::swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value , class _Hash , class _Pred , class _Alloc >
  void **std::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)

5.595.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_set>.

Definition in file unordered_set.h.

5.596 update_fn_imps.hpp File Reference

5.596.1 Detailed Description

Contains an implementation class for pat_trie_.

Definition in file update_fn_imps.hpp.

5.597 utility File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_UTILITY**

Functions

- template<std::size_t _Int, class _Tp1 , class _Tp2 >
  constexpr tuple_element< _Int,
  std::pair< _Tp1, _Tp2 >
  >::type & **std::get** (std::pair< _Tp1, _Tp2 > &__in) noexcept
- template<std::size_t _Int, class _Tp1 , class _Tp2 >
  constexpr tuple_element< _Int,
  std::pair< _Tp1, _Tp2 >
  >::type && **std::get** (std::pair< _Tp1, _Tp2 > &&__in) noexcept

- template<std::size_t _Int, class _Tp1 , class _Tp2 >
  constexpr const tuple_element
  < _Int, std::pair< _Tp1, _Tp2 >
  >::type & **std::get** (const std::pair< _Tp1, _Tp2 > &__in) noexcept

5.597.1 Detailed Description

This is a Standard C++ Library header.

Definition in file utility.

5.598 valarray File Reference

Classes

- class std::valarray< _Tp >

  Smart array designed to support numeric processing.

Namespaces

- namespace std

Macros

- #define **_DEFINE_BINARY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT**(_Op, _Name)
- #define **_DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT**(_Op, _Name)
- #define **_DEFINE_VALARRAY_UNARY_OPERATOR**(_Op, _Name)
- #define **_GLIBCXX_VALARRAY**

Functions

- template<class _Tp >
  _Tp ∗ std::begin (valarray< _Tp > &__va)
- template<class _Tp >
  const _Tp ∗ std::begin (const valarray< _Tp > &__va)
- template<class _Tp >
  _Tp ∗ std::end (valarray< _Tp > &__va)
- template<class _Tp >
  const _Tp ∗ std::end (const valarray< _Tp > &__va)
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _Constant, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **std::operator!=** (const valarray< _Tp > &__v, const _Tp &__t)

- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _Constant,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **std::operator!=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __not_equal_to, _Tp >
  ::result_type > **std::operator!=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **std::operator%** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **std::operator%** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __modulus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __modulus,
  _Tp >::result_type > **std::operator%** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **std::operator&** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **std::operator&** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_and,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_and, _Tp >
  ::result_type > **std::operator&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **std::operator&&** (const valarray< _Tp > &__v, const _Tp &__t)

- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **std::operator&&** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __logical_and,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __logical_and, _Tp >
  ::result_type > **std::operator&&** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **std::operator*** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **std::operator*** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __multiplies,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __multiplies,
  _Tp >::result_type > **std::operator*** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **std::operator+** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **std::operator+** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __plus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __plus, _Tp >
  ::result_type > **std::operator+** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **std::operator-** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **std::operator-** (const valarray< _Tp > &__v, const _Tp &__t)

- template<typename _Tp >
  _Expr< _BinClos< __minus,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __minus, _Tp >
  ::result_type > **std::operator-** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **std::operator/** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **std::operator/** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __divides,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __divides,
  _Tp >::result_type > **std::operator/** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **std::operator<** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **std::operator<** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __less,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less, _Tp >
  ::result_type > **std::operator<** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **std::operator<<** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **std::operator<<** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __shift_left,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __shift_left,
  _Tp >::result_type > **std::operator<<** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,

_Tp >::result_type > **std::operator<=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **std::operator<=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __less_equal,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __less_equal,
  _Tp >::result_type > **std::operator<=** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **std::operator==** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **std::operator==** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __equal_to,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __equal_to,
  _Tp >::result_type > **std::operator==** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **std::operator>** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **std::operator>** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __greater,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __greater,
  _Tp >::result_type > **std::operator>** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **std::operator>=** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _Constant,
  _ValArray, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >

::result_type > **std::operator>=** (const _Tp &__t, const valarray< _Tp > &__v)

- template<typename _Tp >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _Constant, _Tp, _Tp >
  , typename __fun
  < __greater_equal, _Tp >
  ::result_type > **std::operator>=** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **std::operator>>** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **std::operator>>** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __shift_right,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __shift_right, _Tp >
  ::result_type > **std::operator>>** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **std::operator^** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **std::operator^** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun
  < __bitwise_xor, _Tp >
  ::result_type > **std::operator^** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **std::operator|** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **std::operator|** (const valarray< _Tp > &__v, const _Tp &__t)

- template<typename _Tp >
  _Expr< _BinClos< __bitwise_or,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __bitwise_or,
  _Tp >::result_type > **std::operator|** (const _Tp &__t, const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _BinClos< __logical_or,
  _ValArray, _ValArray, _Tp, _Tp >
  , typename __fun< __logical_or,
  _Tp >::result_type > **std::operator||** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<typename _Tp >
  _Expr< _BinClos< __logical_or,
  _ValArray, _Constant, _Tp, _Tp >
  , typename __fun< __logical_or,
  _Tp >::result_type > **std::operator||** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< __logical_or,
  _Constant, _ValArray, _Tp, _Tp >
  , typename __fun< __logical_or,
  _Tp >::result_type > **std::operator||** (const _Tp &__t, const valarray< _Tp > &__v)

5.598.1 Detailed Description

This is a Standard C++ Library header.

Definition in file valarray.

5.599 valarray_after.h File Reference

Namespaces

- namespace std

Macros

- #define **_DEFINE_EXPR_BINARY_FUNCTION**(_Fun, _UFun)
- #define **_DEFINE_EXPR_BINARY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_EXPR_UNARY_FUNCTION**(_Name, _UName)
- #define **_DEFINE_EXPR_UNARY_OPERATOR**(_Op, _Name)

Functions

- template<class _Dom >
  _Expr< _UnClos< _Abs, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Abs,
  _ValArray, _Tp >, _Tp > **std::abs** (const valarray< _Tp > &__v)

- template<class _Dom >
  _Expr< _UnClos< _Acos, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::acos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Acos,
  _ValArray, _Tp >, _Tp > **std::acos** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Asin, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::asin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Asin,
  _ValArray, _Tp >, _Tp > **std::asin** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Atan, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::atan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Atan,
  _ValArray, _Tp >, _Tp > **std::atan** (const valarray< _Tp > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< _Atan2, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename _Dom1::value_type > **std::atan2** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom >
  _Expr< _BinClos< _Atan2, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **std::atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< _Atan2,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **std::atan2** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< _Atan2, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **std::atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< _Atan2,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **std::atan2** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _BinClos< _Atan2,
  _ValArray, _ValArray, _Tp, _Tp >
  , _Tp > **std::atan2** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

- template<typename _Tp >
  _Expr< _BinClos< _Atan2,
  _ValArray, _Constant, _Tp, _Tp >
  , _Tp > **std::atan2** (const valarray< _Tp > &__v, const _Tp &__t)
- template<typename _Tp >
  _Expr< _BinClos< _Atan2,
  _Constant, _ValArray, _Tp, _Tp >
  , _Tp > **std::atan2** (const _Tp &__t, const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Cos, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::cos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Cos,
  _ValArray, _Tp >, _Tp > **std::cos** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Cosh, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::cosh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Cosh,
  _ValArray, _Tp >, _Tp > **std::cosh** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Exp, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::exp** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Exp,
  _ValArray, _Tp >, _Tp > **std::exp** (const valarray< _Tp > &__v)
- template<typename _Tp >
  _Expr< _UnClos< _Log,
  _ValArray, _Tp >, _Tp > **std::log** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Log, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::log** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _UnClos< _Log10, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::log10** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Log10,
  _ValArray, _Tp >, _Tp > **std::log10** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _Constant,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __not_equal_to, typename
  _Dom::value_type >
  ::result_type > **std::operator!=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __not_equal_to,
  typename _Dom::value_type >
  ::result_type > **std::operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _ValArray,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __not_equal_to, typename
  _Dom::value_type >
  ::result_type > **std::operator!=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos
  < __not_equal_to, _Expr, _Expr,
  _Dom1, _Dom2 >, typename __fun
  < __not_equal_to, typename
  _Dom1::value_type >
  ::result_type > **std::operator!=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos
  < __not_equal_to, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __not_equal_to,
  typename _Dom::value_type >
  ::result_type > **std::operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __modulus,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __modulus,
  typename _Dom1::value_type >
  ::result_type > **std::operator%** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >

_Expr< _BinClos< __modulus,
_Constant, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __modulus,
typename _Dom::value_type >
::result_type > **std::operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __modulus,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __modulus,
  typename _Dom::value_type >
  ::result_type > **std::operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **std::operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_and,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_and, typename
  _Dom::value_type >
  ::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >

_Expr< _BinClos< __bitwise_and,
_ValArray, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun
< __bitwise_and, typename
_Dom::value_type >
::result_type > **std::operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __bitwise_and,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __bitwise_and, typename
  _Dom1::value_type >
  ::result_type > **std::operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __logical_and,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __logical_and, typename
  _Dom1::value_type >
  ::result_type > **std::operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **std::operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __logical_and,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __logical_and, typename
  _Dom::value_type >
  ::result_type > **std::operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >

_Expr< _BinClos< __logical_and,
_Expr, _Constant, _Dom,
typename _Dom::value_type >
, typename __fun
< __logical_and, typename
_Dom::value_type >
::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __multiplies,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __multiplies,
  typename _Dom1::value_type >
  ::result_type > **std::operator∗** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **std::operator∗** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **std::operator∗** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **std::operator∗** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __multiplies,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __multiplies,
  typename _Dom::value_type >
  ::result_type > **std::operator∗** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __plus, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __plus, typename
  _Dom::value_type >
  ::result_type > **std::operator+** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<

typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __plus, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __plus, typename
  _Dom::value_type >
  ::result_type > **std::operator+** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __plus,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __plus,
  typename _Dom::value_type >
  ::result_type > **std::operator+** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __plus, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename __fun< __plus,
  typename _Dom1::value_type >
  ::result_type > **std::operator+** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __plus,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __plus,
  typename _Dom::value_type >
  ::result_type > **std::operator+** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __minus,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __minus,
  typename _Dom::value_type >
  ::result_type > **std::operator-** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __minus,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __minus,
  typename _Dom::value_type >
  ::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >

_Expr< _BinClos< __minus,
_ValArray, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __minus,
typename _Dom::value_type >
::result_type > **std::operator-** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __minus,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __minus,
  typename _Dom1::value_type >
  ::result_type > **std::operator-** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __minus,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __minus,
  typename _Dom::value_type >
  ::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __divides,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __divides,
  typename _Dom1::value_type >
  ::result_type > **std::operator/** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __divides,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **std::operator/** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __divides,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **std::operator/** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __divides,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **std::operator/** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __divides,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __divides,
  typename _Dom::value_type >
  ::result_type > **std::operator/** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __less, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename __fun< __less,
  typename _Dom1::value_type >
  ::result_type > **std::operator<** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __less, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __less, typename
  _Dom::value_type >
  ::result_type > **std::operator<** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __less,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __less,
  typename _Dom::value_type >
  ::result_type > **std::operator<** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __less, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __less, typename
  _Dom::value_type >
  ::result_type > **std::operator<** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __less,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __less,
  typename _Dom::value_type >
  ::result_type > **std::operator<** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __shift_left,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __shift_left,
  typename _Dom1::value_type >
  ::result_type > **std::operator<<** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **std::operator<<** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **std::operator<<** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **std::operator<<** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_left,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __shift_left,
  typename _Dom::value_type >
  ::result_type > **std::operator<<** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __less_equal,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __less_equal,
  typename _Dom1::value_type >
  ::result_type > **std::operator<=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __less_equal,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __less_equal,
  typename _Dom::value_type >
  ::result_type > **std::operator<=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __less_equal,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __less_equal,
  typename _Dom::value_type >
  ::result_type > **std::operator<=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<

typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __less_equal,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __less_equal,
  typename _Dom::value_type >
  ::result_type > **std::operator<=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __less_equal,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __less_equal,
  typename _Dom::value_type >
  ::result_type > **std::operator<=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **std::operator==** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **std::operator==** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __equal_to,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __equal_to,
  typename _Dom::value_type >
  ::result_type > **std::operator==** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __equal_to,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __equal_to,
  typename _Dom1::value_type >
  ::result_type > **std::operator==** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >

_Expr< _BinClos< __equal_to,
_Expr, _Constant, _Dom,
typename _Dom::value_type >
, typename __fun< __equal_to,
typename _Dom::value_type >
::result_type > **std::operator==** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom >
  _Expr< _BinClos< __greater,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **std::operator>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __greater,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **std::operator>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __greater,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **std::operator>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __greater,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __greater,
  typename _Dom1::value_type >
  ::result_type > **std::operator>** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __greater,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __greater,
  typename _Dom::value_type >
  ::result_type > **std::operator>** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  __fun< __greater_equal,
  typename _Dom::value_type >
  ::result_type > **std::operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<

typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _Constant,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __greater_equal, typename
  _Dom::value_type >
  ::result_type > **std::operator>=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _ValArray,
  _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __greater_equal, typename
  _Dom::value_type >
  ::result_type > **std::operator>=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos
  < __greater_equal, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __greater_equal, typename
  _Dom1::value_type >
  ::result_type > **std::operator>=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos
  < __greater_equal, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  __fun< __greater_equal,
  typename _Dom::value_type >
  ::result_type > **std::operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __shift_right,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun
  < __shift_right, typename
  _Dom1::value_type >
  ::result_type > **std::operator>>** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >

_Expr< _BinClos< __shift_right,
_Expr, _Constant, _Dom,
typename _Dom::value_type >
, typename __fun
< __shift_right, typename
_Dom::value_type >
::result_type > **std::operator>>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **std::operator>>** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **std::operator>>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __shift_right,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __shift_right, typename
  _Dom::value_type >
  ::result_type > **std::operator>>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **std::operator^** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **std::operator^** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >

_Expr< _BinClos< __bitwise_xor,
_Expr, _Expr, _Dom1, _Dom2 >
, typename __fun
< __bitwise_xor, typename
_Dom1::value_type >
::result_type > **std::operator^** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **std::operator^** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_xor,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun
  < __bitwise_xor, typename
  _Dom::value_type >
  ::result_type > **std::operator^** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __bitwise_or,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __bitwise_or,
  typename _Dom1::value_type >
  ::result_type > **std::operator|** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **std::operator|** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **std::operator|** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >

_Expr< _BinClos< __bitwise_or,
_Constant, _Expr, typename
_Dom::value_type, _Dom >
, typename __fun< __bitwise_or,
typename _Dom::value_type >
::result_type > **std::operator|** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom >
  _Expr< _BinClos< __bitwise_or,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __bitwise_or,
  typename _Dom::value_type >
  ::result_type > **std::operator|** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __logical_or,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __logical_or,
  typename _Dom::value_type >
  ::result_type > **std::operator||** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< __logical_or,
  _Expr, _Constant, _Dom,
  typename _Dom::value_type >
  , typename __fun< __logical_or,
  typename _Dom::value_type >
  ::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom >
  _Expr< _BinClos< __logical_or,
  _Expr, _ValArray, _Dom,
  typename _Dom::value_type >
  , typename __fun< __logical_or,
  typename _Dom::value_type >
  ::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom >
  _Expr< _BinClos< __logical_or,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename __fun< __logical_or,
  typename _Dom::value_type >
  ::result_type > **std::operator||** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< __logical_or,
  _Expr, _Expr, _Dom1, _Dom2 >
  , typename __fun< __logical_or,
  typename _Dom1::value_type >
  ::result_type > **std::operator||** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom >
  _Expr< _BinClos< _Pow,
  _ValArray, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **std::pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom >
  _Expr< _BinClos< _Pow,
  _Constant, _Expr, typename
  _Dom::value_type, _Dom >
  , typename _Dom::value_type > **std::pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _BinClos< _Pow,
  _ValArray, _Constant, _Tp, _Tp >
  , _Tp > **std::pow** (const valarray< _Tp > &__v, const _Tp &__t)
- template<class _Dom >
  _Expr< _BinClos< _Pow, _Expr,
  _Constant, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **std::pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<typename _Tp >
  _Expr< _BinClos< _Pow,
  _ValArray, _ValArray, _Tp, _Tp >
  , _Tp > **std::pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom >
  _Expr< _BinClos< _Pow, _Expr,
  _ValArray, _Dom, typename
  _Dom::value_type >, typename
  _Dom::value_type > **std::pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1 , class _Dom2 >
  _Expr< _BinClos< _Pow, _Expr,
  _Expr, _Dom1, _Dom2 >
  , typename _Dom1::value_type > **std::pow** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<typename _Tp >
  _Expr< _BinClos< _Pow,
  _Constant, _ValArray, _Tp, _Tp >
  , _Tp > **std::pow** (const _Tp &__t, const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Sin, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::sin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp >
  _Expr< _UnClos< _Sin,
  _ValArray, _Tp >, _Tp > **std::sin** (const valarray< _Tp > &__v)
- template<class _Dom >
  _Expr< _UnClos< _Sinh, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::sinh** (const _Expr< _Dom, typename _Dom::value_type > &__e)

- template< typename _Tp >
  _Expr< _UnClos< _Sinh,
  _ValArray, _Tp >, _Tp > **std::sinh** (const valarray< _Tp > &__v)
- template< typename _Tp >
  _Expr< _UnClos< _Sqrt,
  _ValArray, _Tp >, _Tp > **std::sqrt** (const valarray< _Tp > &__v)
- template< class _Dom >
  _Expr< _UnClos< _Sqrt, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::sqrt** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template< typename _Tp >
  _Expr< _UnClos< _Tan,
  _ValArray, _Tp >, _Tp > **std::tan** (const valarray< _Tp > &__v)
- template< class _Dom >
  _Expr< _UnClos< _Tan, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::tan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template< class _Dom >
  _Expr< _UnClos< _Tanh, _Expr,
  _Dom >, typename
  _Dom::value_type > **std::tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template< typename _Tp >
  _Expr< _UnClos< _Tanh,
  _ValArray, _Tp >, _Tp > **std::tanh** (const valarray< _Tp > &__v)

5.599.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file valarray_after.h.

5.600 valarray_array.h File Reference

Namespaces

- namespace std

Macros

- #define **_DEFINE_ARRAY_FUNCTION**(_Op, _Name)

Functions

- template< typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __a, size_t __n, _Tp ∗__restrict__ __b)
- template< typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __a, size_t __n, size_t __s, _Tp ∗__restrict__ __b)
- template< typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __a, _Tp ∗__restrict__ __b, size_t __n, size_t __s)

- template<typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __src, size_t __n, size_t __s1, _Tp ∗__restrict__ __dst, size_t __s2)
- template<typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __a, const size_t ∗__restrict__ __i, _Tp ∗__restrict__ __b, size_t __n)
- template<typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __a, size_t __n, _Tp ∗__restrict__ __b, const size_t ∗__restrict__ __i)
- template<typename _Tp >
  void **std::__valarray_copy** (const _Tp ∗__restrict__ __src, size_t __n, const size_t ∗__restrict__ __i, _Tp ∗__restrict__ __dst, const size_t ∗__restrict__ __j)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)
- template<typename _Tp >
  void **std::__valarray_copy_construct** (const _Tp ∗__b, const _Tp ∗__e, _Tp ∗__restrict__ __o)
- template<typename _Tp >
  void **std::__valarray_copy_construct** (const _Tp ∗__restrict__ __a, size_t __n, size_t __s, _Tp ∗__restrict__ __o)
- template<typename _Tp >
  void **std::__valarray_copy_construct** (const _Tp ∗__restrict__ __a, const size_t ∗__restrict__ __i, _Tp ∗__restrict__ __o, size_t __n)
- template<typename _Tp >
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::__valarray_default_construct** (_Tp ∗__b, _Tp ∗__e)
- template<typename _Tp >
  void **std::__valarray_destroy_elements** (_Tp ∗__b, _Tp ∗__e)
- template<typename _Tp >
  void **std::__valarray_fill** (_Tp ∗__restrict__ __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::__valarray_fill** (_Tp ∗__restrict__ __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp >
  void **std::__valarray_fill** (_Tp ∗__restrict__ __a, const size_t ∗__restrict__ __i, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, const _Tp &__t)

- template<typename _Tp >
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp >
  void **std::__valarray_fill** (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::__valarray_fill_construct** (_Tp ∗__b, _Tp ∗__e, const _Tp __t)
- void ∗ **std::__valarray_get_memory** (size_t __n)
- template<typename _Tp >
  _Tp ∗__restrict__ **std::__valarray_get_storage** (size_t __n)
- template<typename _Ta >
  _Ta::value_type **std::__valarray_max** (const _Ta &__a)
- template<typename _Ta >
  _Ta::value_type **std::__valarray_min** (const _Ta &__a)
- template<typename _Tp >
  _Tp **std::__valarray_product** (const _Tp ∗__f, const _Tp ∗__l)
- void **std::__valarray_release_memory** (void ∗__p)
- template<typename _Tp >
  _Tp **std::__valarray_sum** (const _Tp ∗__f, const _Tp ∗__l)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)

- template<typename _Tp , class _Dom >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp >
void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)

- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)

- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)

5.600.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file valarray_array.h.

5.601 valarray_array.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_VALARRAY_ARRAY_TCC**

Functions

- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)
- template<typename _Tp , class _Dom >
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp , class _Dom >
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)

- template<typename _Tp , class _Dom >
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp >
  void **std::__valarray_copy** (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp , class _Dom >
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)
- template<typename _Tp , class _Dom >
  void **std::__valarray_copy_construct** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp >
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp >
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)

5.601.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file valarray_array.tcc.

5.602 valarray_before.h File Reference

Namespaces

- namespace std

5.602.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

Definition in file valarray_before.h.

5.603 vector File Reference

Macros

- #define **_GLIBCXX_VECTOR**

5.603.1 Detailed Description

This is a Standard C++ Library header.

Definition in file vector.

5.604 vector File Reference

Classes

- class std::__debug::vector< _Tp, _Allocator >

  Class std::vector with safety/checking/debug instrumentation.
- struct std::hash< __debug::vector< bool, _Alloc > >

  std::hash specialization for vector<bool>.

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_VECTOR**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator!=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator<=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator>** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__debug::operator>=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__debug::swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)

5.604.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

Definition in file debug/vector.

5.605 vector File Reference

Classes

- struct std::hash< __profile::vector< bool, _Alloc > >

  std::hash specialization for vector<bool>.

Namespaces

- namespace std
- namespace std::__profile

Macros

- #define **_GLIBCXX_PROFILE_VECTOR**

Functions

- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator!=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator<** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator<=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator>** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  bool **std::__profile::operator>=** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__profile::swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__profile::swap** (vector< _Tp, _Alloc > &&__lhs, vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp , typename _Alloc >
  void **std::__profile::swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &&__rhs)

5.605.1 Detailed Description

This file is a GNU profile extension to the Standard C++ Library.

Definition in file profile/vector.

5.606 vector.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_VECTOR_TCC**

5.606.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

Definition in file vector.tcc.

5.607 vstring.h File Reference

Classes

- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

  Template class __versa_string.
  Data structure managing sequences of characters and character-like objects.
- struct std::hash< __gnu_cxx::__u16vstring >

  std::hash specialization for __u16vstring.
- struct std::hash< __gnu_cxx::__u32vstring >

  std::hash specialization for __u32vstring.
- struct std::hash< __gnu_cxx::__vstring >

  std::hash specialization for __vstring.
- struct std::hash< __gnu_cxx::__wvstring >

  std::hash specialization for __wvstring.

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator!= (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (const _CharT ∗__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator< (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)

- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , template< typename, typename, typename > class _Base>
  __enable_if< std::__is_char
  < _CharT >::__value, bool >
  ::__type **__gnu_cxx::operator==** (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)

5.607.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Definition in file vstring.h.

5.608 vstring.tcc File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_VSTRING_TCC**

Functions

- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT ∗__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT ∗__rhs)
- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  __versa_string< _CharT,
  _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template< typename _CharT , typename _Traits , typename _Alloc , template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)

5.608.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

Definition in file vstring.tcc.

5.609 vstring_fwd.h File Reference

Classes

- class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >
- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

Template class __versa_string.
Data structure managing sequences of characters and character-like objects.

Namespaces

- namespace __gnu_cxx

Typedefs

- typedef __versa_string< char,
  std::char_traits< char >
  , std::allocator< char >
  , __rc_string_base > **__gnu_cxx::__rc_string**
- typedef __vstring **__gnu_cxx::__sso_string**
- typedef __versa_string
  < char16_t, std::char_traits
  < char16_t >, std::allocator
  < char16_t >, __rc_string_base > **__gnu_cxx::__u16rc_string**
- typedef __u16vstring **__gnu_cxx::__u16sso_string**
- typedef __versa_string< char16_t > **__gnu_cxx::__u16vstring**
- typedef __versa_string
  < char32_t, std::char_traits
  < char32_t >, std::allocator
  < char32_t >, __rc_string_base > **__gnu_cxx::__u32rc_string**
- typedef __u32vstring **__gnu_cxx::__u32sso_string**
- typedef __versa_string< char32_t > **__gnu_cxx::__u32vstring**
- typedef __versa_string< char > **__gnu_cxx::__vstring**
- typedef __versa_string
  < wchar_t, std::char_traits
  < wchar_t >, std::allocator
  < wchar_t >, __rc_string_base > **__gnu_cxx::__wrc_string**
- typedef __wvstring **__gnu_cxx::__wsso_string**
- typedef __versa_string< wchar_t > **__gnu_cxx::__wvstring**

5.609.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

Definition in file vstring_fwd.h.

5.610 vstring_util.h File Reference

Namespaces

- namespace __gnu_cxx

5.610.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

Definition in file vstring_util.h.

5.611 workstealing.h File Reference

Classes

- struct __gnu_parallel::_Job< _DifferenceTp >

  One __job for a certain thread.

Namespaces

- namespace __gnu_parallel

Macros

- #define **_GLIBCXX_JOB_VOLATILE**

Functions

- template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
  _Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

5.611.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing. Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. Journal of the ACM, 46(5):720–748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.

Definition in file workstealing.h.

Index