

Search: Go

Not logged in

Reference <set> multiset erase

register

log in

C++

Information

Tutorials

Reference

Articles

Forum

Reference

C library:

Containers:

<array>

<deque>

<forward_list>

<list>

<map>

<queue>

<set>

<stack>

<unordered_map>

<unordered_set>

<vector>

Input/Output:

Multi-threading:

Other:

<set>

multiset
set

multiset

multiset::multiset
multiset::~multiset

member functions:

multiset::begin

multiset::cbegin

multiset::cend

multiset::clear

multiset::count

multiset::crbegin

multiset::crend

multiset::emplace

multiset::emplace_hint

multiset::empty

multiset::end

multiset::equal_range

multiset::erase

multiset::find

multiset::get_allocator

multiset::insert

multiset::key_comp

multiset::lower_bound

multiset::max_size

multiset::operator=

multiset::rbegin

multiset::rend

multiset::size

multiset::swap

multiset::upper_bound

multiset::value_comp

non-member overloads:

relational operators (multiset)

swap (multiset)

public member function

std::multiset::erase

<set>

C++98

C++11

```
(1) iterator erase (const_iterator position);
(2) size_type erase (const value_type& val);
(3) iterator erase (const_iterator first, const_iterator last);
```

Erase elements

Removes elements from the [multiset](#) container.This effectively reduces the container [size](#) by the number of elements removed, which are destroyed.

The parameters determine the elements removed:

Parameters

position

Iterator pointing to a single element to be removed from the [multiset](#).Member types [iterator](#) and [const_iterator](#) are [bidirectional iterator](#) types that point to elements.

val

Value to be removed from the [multiset](#). All elements with a value equivalent to this are removed from the container.Member type [value_type](#) is the type of the elements in the container, defined in [multiset](#) as an alias of its first template parameter (T).

first, last

Iterators specifying a range within the [multiset](#) container to be removed: [first,last). i.e., the range includes all the elements between *first* and *last*, including the element pointed by *first* but not the one pointed by *last*. Member types [iterator](#) and [const_iterator](#) are [bidirectional iterator](#) types that point to elements.

Return value

For the value-based version (2), the function returns the number of elements erased.

Member type [size_type](#) is an unsigned integral type.

C++98

C++11

The other versions return an iterator to the element that follows the last element removed (or [multiset::end](#), if the last element was removed).Member type [iterator](#) is a [bidirectional iterator](#) type that points to elements.

Example

```
1 // erasing from multiset
2 #include <iostream>
3 #include <set>
4
5 int main ()
6 {
7     std::multiset<int> mymultiset;
8     std::multiset<int>::iterator it;
9
10    // insert some values:
11    mymultiset.insert (40);           // 40
12    for (int i=1; i<7; i++) mymultiset.insert(i*10); // 10 20 30 40 40 50 60
13
14    it=mymultiset.begin();
15    it++;                             // ^
16
17    mymultiset.erase (it);            // 10 30 40 40 50 60
18
19    mymultiset.erase (40);            // 10 30 50 60
20
21    it=mymultiset.find (50);
22    mymultiset.erase ( it, mymultiset.end() ); // 10 30
23
24    std::cout << "mymultiset contains:";
25    for (it=mymultiset.begin(); it!=mymultiset.end(); ++it)
26        std::cout << ' ' << *it;
```

```

27 | std::cout << '\n';
28 |
29 | return 0;
30 | }

```

Output:

```
mymultiset contains: 10 30
```

Complexity

For the first version (`erase(position)`), amortized constant.

For the second version (`erase(val)`), logarithmic in container [size](#), plus linear in the number of elements removed.

For the last version (`erase(first, last)`), linear in the distance between *first* and *last*.

Iterator validity

Iterators, pointers and references referring to elements removed by the function are invalidated.

All other iterators, pointers and references keep their validity.

Data races

The container is modified.

The elements removed are modified. Concurrently accessing other elements is safe, although iterating ranges in the container is not.

Exception safety

Unless the container's [comparison object](#) throws, this function never throws exceptions (no-throw guarantee).

Otherwise, if a single element is to be removed, there are no changes in the container in case of exception (strong guarantee).

Otherwise, the container is guaranteed to end in a valid state (basic guarantee).

If an invalid *position* or range is specified, it causes *undefined behavior*.

See also

multiset::clear	Clear content (public member function)
multiset::insert	Insert element (public member function)
multiset::find	Get iterator to element (public member function)