public member function

std::set::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 from the set container either a single element or a range of elements ([first,last)).

This effectively reduces the container size by the number of elements removed, which are destroyed.

Parameters

position
: Iterator pointing to a single element to be removed from the set.
Member types iterator and const_iterator are bidirectional iterator types that point to elements.

val
: Value to be removed from the set.
Member type value_type is the type of the elements in the container, defined in set as an alias of its first template parameter (T).

first, last
: Iterators specifying a range within the set 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 set::end, if the last element was removed).

Member type iterator is a bidirectional iterator type that points to elements.

Example

```
1  // erasing from set
2  #include <iostream>
3  #include <set>
4
5  int main ()
6  {
7    std::set<int> myset;
8    std::set<int>::iterator it;
9
10   // insert some values:
11   for (int i=1; i<10; i++) myset.insert(i*10);  // 10 20 30 40 50 60 70 80 90
12
13   it = myset.begin();
14   ++it;                                         // "it" points now to 20
15
16   myset.erase (it);
17
18   myset.erase (40);
19
20   it = myset.find (60);
21   myset.erase (it, myset.end());
22
23   std::cout << "myset contains:";
24   for (it=myset.begin(); it!=myset.end(); ++it)
25     std::cout << ' ' << *it;
26   std::cout << '\n';
27
28   return 0;
29 }
```

Output:

```
myset contains: 10 30 50
```

Complexity

For the first version (`erase(position)`), amortized constant.
For the second version (`erase(val)`), logarithmic in container size.
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

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