

C++
Information
Tutorials
Reference
Articles
Forum

Reference
C library:
Containers:
<array>
<deque>
<forward_list>
<list>
<map>
<forward_list>
<queue>
<set>
<stack>
<unordered_map>
<unordered_set>
<vector>
Input/Output:
Multi-threading:
Other:

<map>
map
multimap

map
map::map
map::~map
member functions:
map::at
map::begin
map::cbegin
map::cend
map::clear
map::count
map::crbegin
map::crend
map::emplace
map::emplace_hint
map::empty
map::end
map::equal_range
map::erase
map::find
map::get_allocator
map::insert
map::key_comp
map::lower_bound
map::max_size
map::operator=
map::operator[]
map::rbegin
map::rend
map::size
map::swap
map::upper_bound
map::value_comp
non-member overloads:
relational operators (map)
swap (map)

SMARTBEAR

Superior Test Automation for Everyone

UTS

UTS POSTGRAD EXPO

16 APRIL

REGISTER NOW

22095

public member function

std::map::map

<map>

C++98 C++11 C++14

empty (1) explicit map (const key_compare& comp = key_compare(), const allocator_type& alloc = allocator_type());

range (2) template <class InputIterator> map (InputIterator first, InputIterator last, const key_compare& comp = key_compare(), const allocator_type& alloc = allocator_type());

copy (3) map (const map& x);

Construct map

Constructs a map container object, initializing its contents depending on the constructor version used:

C++98 C++11

(1) empty container constructor (default constructor) Constructs an empty container, with no elements.

(2) range constructor Constructs a container with as many elements as the range [first, last), with each element constructed from its corresponding element in that range.

(3) copy constructor Constructs a container with a copy of each of the elements in x.

The container keeps an internal copy of alloc and comp, which are used to allocate storage and to sort the elements throughout its lifetime.

The copy constructor (3) creates a container that keeps and uses copies of x's allocator and comparison object.

The storage for the elements is allocated using this internal allocator.

The elements are sorted according to the comparison object. If more than one element with equivalent keys is passed to the constructor, only the first one is preserved.

Parameters

comp

Binary predicate that, taking two element keys as argument, returns true if the first argument goes before the second argument in the strict weak ordering it defines, and false otherwise.

This shall be a function pointer or a function object.

Member type key_compare is the internal comparison object type used by the container, defined in map as an alias of its third template parameter (Compare).

If key_compare uses the default less (which has no state), this parameter is not relevant.

alloc

Allocator object.

The container keeps and uses an internal copy of this allocator.

Member type allocator_type is the internal allocator type used by the container, defined in map as an alias of its fourth template parameter (Alloc).

If allocator_type is an instantiation of the default allocator (which has no state), this parameter is not relevant.

first, last

Input iterators to the initial and final positions in a range. The range used is [first, last), which includes all the elements between first and last, including the element pointed by first but not the element pointed by last.

The function template argument InputIterator shall be an input iterator type that points to elements of a type from which value_type objects can be constructed (in map, value_type is an alias of pair<const key_type, mapped_type>)

x

Another map object of the same type (with the same class template arguments Key, T, Compare and Alloc), whose contents are either copied or acquired.

il

An initializer_list object.

These objects are automatically constructed from initializer list declarators.

Member type value_type is the type of the elements in the container, defined in map as an alias of pair<const key_type, mapped_type> (see map types).

Example

```
1 // constructing maps
2 #include <iostream>
3 #include <map>
4
5 bool fncmp (char lhs, char rhs) {return lhs<rhs;}
6
7 struct classcomp {
8     bool operator() (const char& lhs, const char& rhs) const
9     {return lhs<rhs;}
10 };
11
12 int main ()
13 {
14     std::map<char,int> first;
15
16     first['a']=10;
17     first['b']=30;
18     first['c']=50;
19     first['d']=70;
20
21     std::map<char,int> second (first.begin(),first.end());
22
23     std::map<char,int> third (second);
24 }
```

```
25 std::map<char,int,classcomp> fourth;           // class as Compare
26
27 bool(*fn_pt)(char,char) = fncomp;
28 std::map<char,int,bool(*) (char,char)> fifth (fn_pt); // function pointer as Compare
29
30 return 0;
31 }
```

The code does not produce any output, but demonstrates some ways in which a `map` container can be constructed.

Complexity

Constant for the *empty constructors (1)*, and for the *move constructors (4)* (unless *alloc* is different from *x*'s allocator). For all other cases, linear in the distance between the iterators (copy constructions) if the elements are already sorted according to the same criterion. For unsorted sequences, linearithmic ($N \cdot \log N$) in that distance (sorting,copy constructions).

Iterator validity

The *move constructors (4)*, invalidate all iterators, pointers and references related to *x* if the elements are moved.

Data races


All copied elements are accessed.
The *move constructors (4)* modify *x*.

Exception safety

Strong guarantee: no effects in case an exception is thrown.
If `allocator_traits::construct` is not supported with the appropriate arguments for the element constructions, or if the range specified by `[first,last)` is not valid, it causes *undefined behavior*.

See also

map::operator=	Copy container content (public member function)
map::insert	Insert elements (public member function)



Rock'n Dreams
Perfume

Shop Now

Perry Ellis M
Cologne

Shop Now

Eternity Sum...
Perfume

Shop Now