

Search:  

Not logged in

Reference &lt;vector&gt; vector operator=

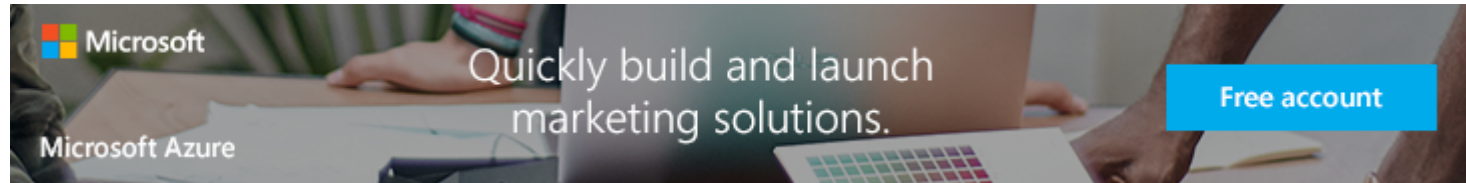
[register](#)[log in](#)

C++
<a href="#">Information</a>
<a href="#">Tutorials</a>
<a href="#">Reference</a>
<a href="#">Articles</a>
<a href="#">Forum</a>

Reference
<i>C library:</i>
<i>Containers:</i>
<a href="#">&lt;array&gt;</a>
<a href="#">&lt;deque&gt;</a>
<a href="#">&lt;forward_list&gt;</a>
<a href="#">&lt;list&gt;</a>
<a href="#">&lt;map&gt;</a>
<a href="#">&lt;queue&gt;</a>
<a href="#">&lt;set&gt;</a>
<a href="#">&lt;stack&gt;</a>
<a href="#">&lt;unordered_map&gt;</a>
<a href="#">&lt;unordered_set&gt;</a>
<a href="#">&lt;vector&gt;</a>
<i>Input/Output:</i>
<i>Multi-threading:</i>
<i>Other:</i>

<vector>
<a href="#">vector</a>
<a href="#">vector&lt;bool&gt;</a>

vector
<a href="#">vector::vector</a>
<a href="#">vector::~~vector</a>
<b>member functions:</b>
<a href="#">vector::assign</a>
<a href="#">vector::at</a>
<a href="#">vector::back</a>
<a href="#">vector::begin</a>
<a href="#">vector::capacity</a>
<a href="#">vector::cbegin</a>
<a href="#">vector::cend</a>



public member function

**std::vector::operator=**

&lt;vector&gt;

[C++98](#) [C++11](#)

```

copy (1)  vector& operator= (const vector& x);
move (2)  vector& operator= (vector&& x);
initializer list (3) vector& operator= (initializer_list<value_type> il);

```

**Assign content**

Assigns new contents to the container, replacing its current contents, and modifying its [size](#) accordingly.

[C++98](#) [C++11](#)

The *copy assignment* (1) copies all the elements from *x* into the container (with *x* preserving its contents).

The *move assignment* (2) moves the elements of *x* into the container (*x* is left in an unspecified but valid state).

The *initializer list assignment* (3) copies the elements of *il* into the container.

The container preserves its [current allocator](#), except if the [allocator traits](#) indicate that *x*'s allocator should [propagate](#). This [allocator](#) is used (through its [traits](#)) to [allocate](#) and [deallocate](#) storage if a reallocation happens, and to [construct](#) or [destroy](#) elements, if needed.

Any elements held in the container before the call are either *assigned to* or *destroyed*.

**Parameters***x*

A [vector](#) object of the same type (i.e., with the same template parameters, *T* and *Alloc*).

*il*

An [initializer\\_list](#) object. The compiler will automatically construct such objects from *initializer list* declarators. Member type `value_type` is the type of the elements in the container, defined in [vector](#) as an alias of its first template parameter (*T*).

[vector::clear](#)  
[vector::cbegin](#)  
[vector::crend](#)  
[vector::data](#)  
[vector::emplace](#)  
[vector::emplace\\_back](#)  
[vector::empty](#)  
[vector::end](#)  
[vector::erase](#)  
[vector::front](#)  
[vector::get\\_allocator](#)  
[vector::insert](#)  
[vector::max\\_size](#)  
[vector::operator=](#)  
[vector::operator\[\]](#)  
[vector::pop\\_back](#)  
[vector::push\\_back](#)  
[vector::rbegin](#)  
[vector::rend](#)  
[vector::reserve](#)  
[vector::resize](#)  
[vector::shrink\\_to\\_fit](#)  
[vector::size](#)  
[vector::swap](#)  
**non-member overloads:**  
[relational operators \(vector\)](#)  
[swap \(vector\)](#)

## Return value

\*this

## Example

```

1 // vector assignment
2 #include <iostream>
3 #include <vector>
4
5 int main ()
6 {
7     std::vector<int> foo (3,0);
8     std::vector<int> bar (5,0);
9
10    bar = foo;
11    foo = std::vector<int>();
12
13    std::cout << "Size of foo: " << int(foo.size()) << '\n';
14    std::cout << "Size of bar: " << int(bar.size()) << '\n';
15    return 0;
16 }
```

Output:

```

Size of foo: 0
Size of bar: 3
```

## Complexity

Linear in [size](#).

## Iterator validity

All iterators, references and pointers related to this container before the call are invalidated.

In the *move assignment*, iterators, pointers and references referring to elements in *x* are also invalidated.

## Data races

All copied elements are accessed.

The *move assignment* (2) modifies *x*.

The container and all its elements are modified.

## Exception safety

**Basic guarantee:** if an exception is thrown, the container is in a valid state.

If `allocator_traits::construct` is not supported with the appropriate arguments for the element constructions, or if `value_type` is not [copy assignable](#) (or [move assignable](#) for (2)), it causes *undefined behavior*.

### See also

<b>vector::assign</b>	Assign vector content ( <a href="#">public member function</a> )
-----------------------	------------------------------------------------------------------

---

[Home page](#) | [Privacy policy](#)

© cplusplus.com, 2000-2017 - All rights reserved - v3.1

[Spotted an error? contact us](#)