Search: Go

Not logged in

Reference <vector> vector **operator=**

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:*

**<vector>**

- vector
- vector<bool>

**vector**

- vector::vector
- vector::~vector
- *member functions:*
  - vector::assign
  - vector::at
  - vector::back
  - vector::begin
  - vector::capacity
  - vector::cbegin
  - vector::cend

public member function

# std::vector::operator=

<vector>

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`).

## Return value

`*this`

## Example

```
// vector assignment
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> foo (3,0);
  std::vector<int> bar (5,0);

  bar = foo;
  foo = std::vector<int>();

  std::cout << "Size of foo: " << int(foo.size()) << '\n';
  std::cout << "Size of bar: " << int(bar.size()) << '\n';
  return 0;
}
```

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

| | |
|---|---|
| **vector::assign** | Assign vector content (public member function ) |

Home page | Privacy policy
© cplusplus.com, 2000-2017 - All rights reserved - *v3.1*
Spotted an error? contact us