---

### 8.10. Allocator Support

All STL containers can be used with a special memory model that is defined by an allocator object (see Chapter 19 for details). This section describes the members for allocator support.

## 8.10.1. Fundamental Allocator Members

### *container*::allocator_type

- The type of the allocator.
- Provided by vector, deque, list, forward list, set, multiset, map, multimap, unordered set, unordered multiset, unordered map, unordered multimap, string.

### `allocator_type` **container::get_allocator** `() const`

- Returns the memory model of the container.
- Provided by vector, deque, list, forward list, set, multiset, map, multimap, unordered set, unordered multiset, unordered map, unordered multimap, string.

## 8.10.2. Constructors with Optional Allocator Parameters

### `explicit` **container**::*container* `(const Allocator&` *alloc*`)`

- Creates a new empty container that uses the memory model *alloc*.
- Provided by vector, deque, list, forward list, set, multiset, map, multimap, unordered set, unordered multiset, unordered map, unordered multimap, string.

### **container**::*container* `(const CompFunc&` *cmpPred*`, const Allocator&` *alloc*`)`

- Creates a new empty container, with *cmpPred* used as the sorting criterion and *alloc* used as memory model.
- The sorting criterion must define a *strict weak ordering* (see Section 7.7, page 314).
- Provided by set, multiset, map, multimap.

Click here to view code image

### **container**::*container* `(size_type` *bnum*`, const Hasher&` *hasher*`, const KeyEqual&` *eqPred*`, const Allocator&` *alloc*`)`

- Creates a new empty container with at least *bnum* buckets, with *hasher* used as hashing function, *eqPred* used as criterion to identify equal values, and *alloc* used as memory model.
- Provided by unordered set, unordered multiset, unordered map, unordered multimap.

### **container**::*container* `(`*initializer-list*`, const Allocator&` *alloc*`)`

- Creates a new container that uses the memory model *alloc* and is initialized by the elements of *initializer-list*.
- Available since C++11.
- Provided by vector, deque, list, forward list, string.

Click here to view code image

### **container**::*container* `(`*initializer-list*`, const CompFunc&` *cmpPred*`, const Allocator&` *alloc*`)`

- Creates a container that has the sorting criterion *cmpPred*, uses the memory model *alloc*, and is initialized by the elements of *initializer-list*.
- The sorting criterion must define a *strict weak ordering* (see Section 7.7, page 314).
- Available since C++11.
- Provided by set, multiset, map, multimap.

Click here to view code image

### **container**::*container* `(`*initializer-list*`, size_type` *bnum*`, const Hasher&` *hasher*`, const KeyEqual&` *eqPred*`, const Allocator&` *alloc*`)`

- Creates a container with at least *bnum* buckets, *hasher* used as hashing function, *eqPred* used as criterion to identify equal values, and *alloc* used as memory model, which is initialized by the elements of *initializer-list*.
- Provided by unordered set, unordered multiset, unordered map, unordered multimap.

Click here to view code image

```
container:: container (const container& c, const Allocator& alloc)
container:: container (container&& c, const Allocator& alloc)
```

- Create a new container that uses the memory model *alloc* and is initialized with copied/moved elements of the existing container *c*.
- Call the copy/move constructor for every element in *c*.
- For the second form, after this call, `c` is valid but has an unspecified value.
- Available since C++11.
- Provided by vector, deque, list, forward list, set, multiset, map, multimap, unordered set, unordered multiset, unordered map, unordered multimap, string.

```
container::container (size_type num, const T& value, const Allocator& alloc)
```

- Creates a container with *num* elements and that uses the memory model *alloc*.
- The elements are created as copies of *value*.
- `T` is the type of the container elements. Note that for strings, *value* is passed by value.
- Provided by vector, deque, list, forward list, string.

```
container::container (InputIterator beg, InputIterator end,
                      const Allocator& alloc)
```

- Creates a container that is initialized by all elements of the range `[` *beg* `,` *end* `)` and uses the memory model *alloc*.
- This function is a member template (see Section 3.2, page 34). Thus, the elements of the source range may have any type convertible into the element type of the container.
- Provided by vector, deque, list, forward list, string.

Click here to view code image

```
container::container (InputIterator beg, InputIterator end,
                      const CompFunc& cmpPred, const Allocator& alloc)
```

- Creates a container that has the sorting criterion *cmpPred*, is initialized by all elements of the range `[` *beg* `,` *end* `)` , and uses the memory model *alloc*.
- This function is a member template (see Section 3.2, page 34). Thus, the elements of the source range may have any type convertible into the element type of the container.
- The sorting criterion must define a *strict weak ordering* (see Section 7.7, page 314).
- Provided by set, multiset, map, multimap.

Click here to view code image

```
container::container (InputIterator beg, InputIterator end,
                      size_type bnum, const Hasher& hasher,
                      const KeyEqual& eqPred, const Allocator& alloc)
```

- Creates a container with at least *bnum* buckets, *hasher* used as hashing function, *eqPred* used as criterion to identify equal values, and *alloc* used as memory model, which is initialized by all elements of the range `[` *beg* `,` *end* `)` .
- Provided by unordered set, unordered multiset, unordered map, unordered multimap.