

**Username:** Pralay Patoria **Book:** The C++ Standard Library: A Tutorial and Reference, Second Edition. No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

---

## 8.5. Direct Element Access

```
reference container::at (size_type idx)
const_reference container::at (size_type idx) const
```

- Return the element with the index *idx* (the first element has index `0` ).
- Passing an invalid index (less than `0` or equal to `size()` or greater than `size()` ) throws an `out_of_range` exception.
- The returned reference may get invalidated due to later modifications or reallocations.
- If you are sure that the index is valid, you can use operator `[]` , which is faster.
- Provided by array, vector, deque, string.

```
T& map::operator at (const key_type& key)
const T& map::operator at (const key_type& key) const
```

- Return the corresponding value to *key* in a map.
- Throw an `out_of_range` exception if no element with a key equal to *key* exists.
- Available since C++11.
- Provided by map, unordered map.

```
reference container::operator [] (size_type idx)
const_reference container::operator [] (size_type idx) const
```

- Both return the element with the index *idx* (the first element has index `0` ).
- Passing an invalid index (less than `0` or equal to `size()` or greater than `size()` ) results in undefined behavior. Thus, the caller must ensure that the index is valid; otherwise, `at()` should be used.
- The returned reference may get invalidated due to later modifications or reallocations.
- Provided by array, vector, deque, string.

```
T& map::operator [] (const key_type& key)
T& map::operator [] (key_type&& key)
```

- Operator `[]` for associative arrays.
- Return the corresponding value to *key* in a map.
- If no element with a key equal to *key* exists, these operations *create* a new element automatically with this key (copied or moved) and a value that is initialized by the default constructor of the value type. Thus, you can't have an invalid index (only wrong behavior). [See Section 6.2.4, page 185](#), and [Section 7.8.3, page 344](#), for details.
- With the second form, the state of *key* is undefined afterward (this form provides move semantics for the case that the key doesn't exist yet).
- The first form is equivalent to:

```
(*(insert(make_pair(key, T()))).first)).second
```

- The second form is available since C++11.
- Provided by map, unordered map.

```
reference container::front ()
const_reference container::front () const
```

- Both return the first element (the element with index `0` ).
- The caller must ensure that the container contains an element ( `size()>0` ); otherwise, the behavior is undefined.
- For strings, it is provided since C++11.
- Provided by array, vector, deque, list, forward list, string.

```
reference container::back ()
const_reference container::back () const
```

- Both return the last element (the element with index `size()-1` ).
- The caller must ensure that the container contains an element ( `size()>0` ); otherwise, the behavior is undefined.
- For strings, it is provided since C++11.
- Provided by `array`, `vector`, `deque`, `list`, `string`.

```
T* container::data ()  
const T* container::data () const
```

- Both return an ordinary C-style array with all elements (that is, a pointer to the first element).
- This function is provided to pass the elements of the array to C-style interfaces.
- For strings, only the second form is provided.
- For arrays and vectors, available since C++11.
- Provided by `array`, `vector`, `string`.