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
vector::clear
vector::crbegin
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)

public member function

std::vector::at

<vector>

```
      reference at (size_type n);
const_reference at (size_type n) const;
```

Access element

Returns a reference to the element at position *n* in the vector.

The function automatically checks whether *n* is within the bounds of valid elements in the vector, throwing an out_of_range exception if it is not (i.e., if *n* is greater than, or equal to, its `size`). This is in contrast with member operator[], that does not check against bounds.

Parameters

n

Position of an element in the container.
If this is greater than, or equal to, the vector size, an exception of type out_of_range is thrown.
Notice that the first element has a position of `0` (not `1`).
Member type `size_type` is an unsigned integral type.

Return value

The element at the specified position in the container.

If the vector object is const-qualified, the function returns a `const_reference`. Otherwise, it returns a `reference`.

Member types `reference` and `const_reference` are the reference types to the elements of the container (see vector member types).

Example

```
// vector::at
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector (10);   // 10 zero-initialized ints

  // assign some values:
  for (unsigned i=0; i<myvector.size(); i++)
    myvector.at(i)=i;

  std::cout << "myvector contains:";
  for (unsigned i=0; i<myvector.size(); i++)
    std::cout << ' ' << myvector.at(i);
  std::cout << '\n';

  return 0;
}
```

Output:

```
myvector contains: 0 1 2 3 4 5 6 7 8 9
```

Complexity

Constant.

Iterator validity

No changes.

Data races

The container is accessed (neither the const nor the non-const versions modify the container).
The reference returned can be used to access or modify elements. Concurrently accessing or modifying different elements is safe.

Exception safety

Strong guarantee: if an exception is thrown, there are no changes in the container.
It throws out_of_range if *n* is out of bounds.

See also

vector::operator[]	Access element (public member function)
vector::front	Access first element (public member function)
vector::back	Access last element (public member function)