# std::vector<bool>

Defined in header `<vector>`

```
template<class Allocator>
class vector<bool, Allocator>;
```

`std::vector<bool>` is a space-efficient specialization of `std::vector` for the type `bool`.

The manner in which `std::vector<bool>` is made space efficient (as well as whether it is optimized at all) is implementation defined. One potential optimization involves coalescing vector elements such that each element occupies a single bit instead of `sizeof(bool)` bytes.

`std::vector<bool>` behaves similarly to `std::vector`, but in order to be space efficient, it:

- Does not necessarily store its elements as a contiguous array (so `&v[0] + n != &v[n]`)
- Exposes class `std::vector<bool>::reference` as a method of accessing individual bits. In particular, objects of this class are returned by `operator[]` by value.
- Does not use `std::allocator_traits::construct` to construct bit values.

## Member types

| Member type | Definition |
|---|---|
| `value_type` | `bool` |
| `allocator_type` | `Allocator` |
| `size_type` | implementation-defined |
| `difference_type` | implementation-defined |
| **`reference`** | proxy class representing a reference to a single bool (class) |
| `const_reference` | `bool` |
| `pointer` | implementation-defined |
| `const_pointer` | implementation-defined |
| `iterator` | implementation-defined |
| `const_iterator` | implementation-defined |
| `reverse_iterator` | `std::reverse_iterator<iterator>` |
| `const_reverse_iterator` | `std::reverse_iterator<const_iterator>` |

## Member functions

| | |
|---|---|
| (constructor) | constructs the `vector` (public member function of `std::vector`) |
| (destructor) | destructs the `vector` (public member function of `std::vector`) |
| **`operator=`** | assigns values to the container (public member function of `std::vector`) |
| **`assign`** | assigns values to the container (public member function of `std::vector`) |
| **`get_allocator`** | returns the associated allocator (public member function of `std::vector`) |

**Element access**

| | |
|---|---|
| **`at`** | access specified element with bounds checking (public member function of `std::vector`) |
| **`operator[]`** | access specified element (public member function of `std::vector`) |
| **`front`** | access the first element |

| | |
|---|---|
| | (public member function of `std::vector`) |
| **`back`** | access the last element<br>(public member function of `std::vector`) |

**Iterators**

| | |
|---|---|
| **`begin`**<br>**`cbegin`** | returns an iterator to the beginning<br>(public member function of `std::vector`) |
| **`end`**<br>**`cend`** | returns an iterator to the end<br>(public member function of `std::vector`) |
| **`rbegin`**<br>**`crbegin`** | returns a reverse iterator to the beginning<br>(public member function of `std::vector`) |
| **`rend`**<br>**`crend`** | returns a reverse iterator to the end<br>(public member function of `std::vector`) |

**Capacity**

| | |
|---|---|
| **`empty`** | checks whether the container is empty<br>(public member function of `std::vector`) |
| **`size`** | returns the number of elements<br>(public member function of `std::vector`) |
| **`max_size`** | returns the maximum possible number of elements<br>(public member function of `std::vector`) |
| **`reserve`** | reserves storage<br>(public member function of `std::vector`) |
| **`capacity`** | returns the number of elements that can be held in currently allocated storage<br>(public member function of `std::vector`) |

**Modifiers**

| | |
|---|---|
| **`clear`** | clears the contents<br>(public member function of `std::vector`) |
| **`insert`** | inserts elements<br>(public member function of `std::vector`) |
| **`emplace`** (since C++14) | constructs element in-place<br>(public member function of `std::vector`) |
| **`erase`** | erases elements<br>(public member function of `std::vector`) |
| **`push_back`** | adds elements to the end<br>(public member function of `std::vector`) |
| **`emplace_back`** (C++14) | constructs elements in-place at the end<br>(public member function of `std::vector`) |
| **`pop_back`** | removes the last element<br>(public member function of `std::vector`) |
| **`resize`** | changes the number of elements stored<br>(public member function of `std::vector`) |
| **`swap`** | swaps the contents<br>(public member function of `std::vector`) |

**`vector<bool>` specific modifiers**

| | |
|---|---|
| **`flip`** | flips all the bits<br>(public member function) |
| **`swap`** [static] | swaps two `std::vector<bool>::reference`s<br>(public static member function) |

## Non-member functions

| | |
|---|---|
| **operator==**<br>**operator!=**<br>**operator<**<br>**operator<=**<br>**operator>**<br>**operator>=** | lexicographically compares the values in the vector<br>(function template) |
| **std::swap**(std::vector) | specializes the `std::swap` algorithm<br>(function template) |

## Helper classes

| | |
|---|---|
| **std::hash**<std::vector<bool>> (C++11) | hash support for `std::vector<bool>`<br>(class template specialization) |

## Notes

If the size of the bitset is known at compile time, `std::bitset` may be used, which offers a richer set of member functions. In addition, boost::dynamic_bitset (http%3A//www.boost.org/doc/libs/release/libs/dynamic_bitset/dynamic_bitset.html) exists as an alternative to `std::vector<bool>`.

Since its representation may by optimized, `std::vector<bool>` does not necessarily meet all `Container` or `SequenceContainer` requirements. For example, because `std::vector<bool>::iterator` is implementation-defined, it may not satisfy the `ForwardIterator` requirement. Use of algorithms such as `std::search` that require `ForwardIterator`s may result in either compile-time or run-time errors (http%3A//www.boost.org/doc/libs/1_52_0/libs/dynamic_bitset/dynamic_bitset.html#rationale) .

Retrieved from "http://en.cppreference.com/mwiki/index.php?title=cpp/container/vector_bool&oldid=78413"

# std::hash (std::vector<bool>)

```
template <class Allocator> struct hash<vector<bool, Allocator>>;
```
(since C++11)

The template specialization of `std::hash` for `std::vector<bool>` allows users to obtain hashes of objects of type `std::vector<bool>`.

## Example

This section is incomplete
Reason: no example

## See also

| | |
|---|---|
| **hash** (C++11) | hash function object (class template) |

Retrieved from "http://en.cppreference.com/mwiki/index.php?title=cpp/container/vector_bool/hash&oldid=63943"

# std::vector<bool>::flip

Defined in header `<vector>`

```
void flip();
```

Toggles each `bool` in the vector (replaces with its opposite value).

## Parameters

(none)

## Return value

(none)

## See also

| | |
|---|---|
| **`operator[]`** | access specified element<br>(public member function of `std::vector`) |
| **`flip`** | toggles the values of bits<br>(public member function of `std::bitset`) |

Retrieved from "http://en.cppreference.com/mwiki/index.php?title=cpp/container/vector_bool/flip&oldid=57332"

## std::vector<bool>::swap

Defined in header `<vector>`

```
static void swap(reference x, reference y);
```

Swaps the contents of `x` and `y`.

### Parameters

| | | | |
|---|---|---|---|
| **x** | - | `std::vector<bool>::reference` | value to swap with `y` |
| **y** | - | `std::vector<bool>::reference` | value to swap with `x` |

### Return value

(none)

### See also

| | |
|---|---|
| **`reference`** | proxy class representing a reference to a single bool (class) |
| **`std::swap`**`(std::vector)` | specializes the `std::swap` algorithm (function template) |

Retrieved from "http://en.cppreference.com/mwiki/index.php?title=cpp/container/vector_bool/swap&oldid=57337"

# std::vector<bool>::reference

```
class reference;
```

The `std::vector<bool>` specialization defines `std::vector<bool>::reference` as a publicly-accessible nested class. `std::vector<bool>::reference` proxies the behavior of references to a single bit in `std::vector<bool>`.

The primary use of `std::vector<bool>::reference` is to provide an l-value that can be returned from `operator[]`.

Any reads or writes to a vector that happen via a `std::vector<bool>::reference` potentially read or write to the entire underlying vector.

## Member functions

| | |
|---|---|
| (constructor) | constructs the reference. Accessible only to `std::vector<bool>` itself (public member function) |
| (destructor) | destroys the reference (public member function) |
| **operator=** | assigns a `bool` to the referenced bit (public member function) |
| **operator bool** | returns the referenced bit (public member function) |
| **flip** | flips the referenced bit (public member function) |

## std::vector<bool>::~reference

```
~reference()
```

Destroys the reference.

## std::vector<bool>::reference::operator=

```
reference& operator=( bool x );
reference& operator=( const reference& x );
```

Assigns a value to the referenced bit.

### Parameters

**x** - value to assign

### Return value

`*this`

### Exceptions

| | |
|---|---|
| (none) | (until C++11) |

| | | |
|---|---|---|
| `noexcept` specification: | `noexcept` | (since C++11) |

### std::vector<bool>::reference::**operator bool**

```
operator bool() const;
```

Returns the value of the referenced bit.

#### Parameters

(none)

#### Return value

The referenced bit.

#### Exceptions

| | | |
|---|---|---|
| (none) | | (until C++11) |
| `noexcept` specification: | `noexcept` | (since C++11) |

### std::vector<bool>::reference::**flip**

```
void flip();
```

Inverts the referenced bit.

#### Parameters

(none)

#### Return value

(none)

#### Exceptions

| | | |
|---|---|---|
| (none) | | (until C++11) |
| `noexcept` specification: | `noexcept` | (since C++11) |

## See also

| | |
|---|---|
| **`operator[]`** | access specified element<br>(public member function of `std::vector`) |
| **`swap`** [static] | swaps two `std::vector<bool>::references`<br>(public static member function) |

Retrieved from "http://en.cppreference.com/mwiki/index.php?title=cpp/container/vector_bool/reference&oldid=71843"