**vector<int> v(3); // Declare a vector of 3 elements.**

**v[0] = 7;**

**v[1] = v[0] + 3;**

**v[2] = v[0] + v[1]; // v[0] == 7, v[1] == 10, v[2] == 17**

by using the [reverse](https://www.sgi.com/tech/stl/reverse.html) algorithm.

**Reverse (v.begin(), v.end()); // v[0] == 17, v[1] == 10, v[2] == 7**

reverse can be used not only to reverse elements in vectors, but also to reverse elements in lists, and even elements in C arrays. The following program is also valid.

**double A[6] = { 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 };**

**reverse(A, A + 6);**

**for (int i = 0; i < 6; ++i)**

**cout << "A[" << i << "] = " << A[i];**

**// Add some elements to myIntVector**

**myIntVector.push\_back(1);**

**myIntVector.push\_back(4);**

**myIntVector.push\_back(8);**

**for(myIntVectorIterator = myIntVector.begin();myIntVectorIterator !=myIntVector.end(); myIntVectorIterator++)**

**{**

**cout<<\*myIntVectorIterator<<" ";**

**//Should output 1 4 8**

**}**

**Vector Class**

**Library: #include<vector>**

**Syntax: vector <int>numbers;**

**Member Types:**

**Member Functions:**

[**(HYPERLINK "http://www.cplusplus.com/reference/vector/vector/vector/"constructorHYPERLINK "http://www.cplusplus.com/reference/vector/vector/vector/")**](http://www.cplusplus.com/reference/vector/vector/vector/): Construct vector (public member function

[**(HYPERLINK "http://www.cplusplus.com/reference/vector/vector/~vector/"destructorHYPERLINK "http://www.cplusplus.com/reference/vector/vector/~vector/")**](http://www.cplusplus.com/reference/vector/vector/~vector/) : Vector destructor (public member function )

**Operator:** Assign content (public member function )

**Iterators:**

[**begin**](http://www.cplusplus.com/reference/vector/vector/begin/): Return iterator to beginning (public member function )

[**end**](http://www.cplusplus.com/reference/vector/vector/end/) :Return iterator to end (public member function )

[**rbegin**](http://www.cplusplus.com/reference/vector/vector/rbegin/): Return reverse iterator to reverse beginning (public member function )

[**rend**](http://www.cplusplus.com/reference/vector/vector/rend/): Return reverse iterator to reverse end (public member function )\

[**cbeginHYPERLINK "http://www.cplusplus.com/reference/vector/vector/cbegin/"**](http://www.cplusplus.com/reference/vector/vector/cbegin/) Return const\_iterator to beginning (public member function )

[**cendHYPERLINK "http://www.cplusplus.com/reference/vector/vector/cend/"**](http://www.cplusplus.com/reference/vector/vector/cend/) :Return const\_iterator to end (public member function )

[**crbeginHYPERLINK "http://www.cplusplus.com/reference/vector/vector/crbegin/"**](http://www.cplusplus.com/reference/vector/vector/crbegin/) :Return const\_reverse\_iterator to reverse beginning (public member function )

[**crendHYPERLINK "http://www.cplusplus.com/reference/vector/vector/crend/"**](http://www.cplusplus.com/reference/vector/vector/crend/) :Return const\_reverse\_iterator to reverse end (public member function )

**Capacity**:

[**size**](http://www.cplusplus.com/reference/vector/vector/size/) :Return size (public member function )

[**max\_size**](http://www.cplusplus.com/reference/vector/vector/max_size/): Return maximum size (public member function )

[**resize**](http://www.cplusplus.com/reference/vector/vector/resize/): Change size (public member function )

[**capacity**](http://www.cplusplus.com/reference/vector/vector/capacity/): Return size of allocated storage capacity (public member function )

[**empty**](http://www.cplusplus.com/reference/vector/vector/empty/): Test whether vector is empty (public member function )

[**reserve**](http://www.cplusplus.com/reference/vector/vector/reserve/): Request a change in capacity (public member function )

[**shrink\_to\_HYPERLINK "http://www.cplusplus.com/reference/vector/vector/shrink\_to\_fit/"fitHYPERLINK "http://www.cplusplus.com/reference/vector/vector/shrink\_to\_fit/"**](http://www.cplusplus.com/reference/vector/vector/shrink_to_fit/): Shrink to fit (public member function )

**Element access**:

[**operator[HYPERLINK "http://www.cplusplus.com/reference/vector/vector/operator%5b%5d/"]**](http://www.cplusplus.com/reference/vector/vector/operator%5b%5d/):Access element (public member function )

[**at**](http://www.cplusplus.com/reference/vector/vector/at/): Access element (public member function )

[**front**](http://www.cplusplus.com/reference/vector/vector/front/): Access first element (public member function )

[**back**](http://www.cplusplus.com/reference/vector/vector/back/): Access last element (public member function )

[**dataHYPERLINK "http://www.cplusplus.com/reference/vector/vector/data/":HYPERLINK "http://www.cplusplus.com/reference/vector/vector/data/"**](http://www.cplusplus.com/reference/vector/vector/data/) Access data (public member function )

**Modifiers**:

[**assign**](http://www.cplusplus.com/reference/vector/vector/assign/): Assign vector content (public member function )

[**push\_back**](http://www.cplusplus.com/reference/vector/vector/push_back/): Add element at the end (public member function )

[**pop\_back**](http://www.cplusplus.com/reference/vector/vector/pop_back/): Delete last element (public member function )

[**insert**](http://www.cplusplus.com/reference/vector/vector/insert/): Insert elements (public member function )

[**erase**](http://www.cplusplus.com/reference/vector/vector/erase/) :Erase elements (public member function )

[**swap**](http://www.cplusplus.com/reference/vector/vector/swap/) Swap content (public member function )

[**clear**](http://www.cplusplus.com/reference/vector/vector/clear/) :Clear content (public member function )

[**emplaceHYPERLINK "http://www.cplusplus.com/reference/vector/vector/emplace/"**](http://www.cplusplus.com/reference/vector/vector/emplace/) :Construct and insert element (public member function )

[**emplace\_HYPERLINK "http://www.cplusplus.com/reference/vector/vector/emplace\_back/"backHYPERLINK "http://www.cplusplus.com/reference/vector/vector/emplace\_back/"**](http://www.cplusplus.com/reference/vector/vector/emplace_back/): Construct and insert element at the end (public member function )

allocator (public member function )

**Non Member Function overloads:**

[**relationalHYPERLINK "http://www.cplusplus.com/reference/vector/vector/operators/" operators**](http://www.cplusplus.com/reference/vector/vector/operators/): Relational operators for vector (function template )

[**swap**](http://www.cplusplus.com/reference/vector/vector/swap-free/) :Exchange contents of vectors (function template )

[Member Functions](javascript:void(0))

|  |  |
| --- | --- |
| [assign](https://msdn.microsoft.com/en-us/library/azbhc96f.aspx) | Replaces the specified elements with copies of the new elements. |
| [at](https://msdn.microsoft.com/en-us/library/c6kkh778.aspx) | Returns a reference to the element at a specified location in the vector and throws out\_of\_range  exception if the index is less than zero or greater than or equal to **size()**. |
| [back](https://msdn.microsoft.com/en-us/library/0532x4xk.aspx) | Returns a reference to the last element of the vector. |
| [begin](https://msdn.microsoft.com/en-us/library/1sc2e041.aspx) | Returns a random-access iterator to the first element in the vector. |
| [capacity](https://msdn.microsoft.com/en-us/library/azcztabt.aspx) | Returns the number of elements that the vector could contain without allocating more storage. |
| [cbegin](https://msdn.microsoft.com/en-us/library/dd647617.aspx) | Returns a random-access const iterator to the first element in the vector. |
| [cend](https://msdn.microsoft.com/en-us/library/dd647622.aspx) | Returns a random-access const iterator that points to one pastj the end of the vector. |
| [crbegin](https://msdn.microsoft.com/en-us/library/dd647615.aspx) | Returns a const reverse\_iterator to the last element in a vector. |
| [crend](https://msdn.microsoft.com/en-us/library/dd647621.aspx) | Returns a const reverse\_iterator that points to one before the first element in the vector. |
| [clear](https://msdn.microsoft.com/en-us/library/fs5a18ce.aspx) | Erases the elements of the vector. |
| [data](https://msdn.microsoft.com/en-us/library/dd647618.aspx) | Returns a pointer to the first element in the vector. |
| [emplace](https://msdn.microsoft.com/en-us/library/dd647616.aspx) | Inserts an element constructed in place into the vector at a specified position. |
| [emplace\_back](https://msdn.microsoft.com/en-us/library/dd647620.aspx) | Adds an element constructed in place to the end of the vector. |
| [empty](https://msdn.microsoft.com/en-us/library/86968s02.aspx) | Tests if the vector contains any elements. |
| [end](https://msdn.microsoft.com/en-us/library/txa4wa2y.aspx) | Returns a random-access iterator that points to one past the last element of the vector. |
| [erase](https://msdn.microsoft.com/en-us/library/ceh559x2.aspx) | Removes an element or a range of elements in a vector from specified positions. |
| [front](https://msdn.microsoft.com/en-us/library/0z70c7a5.aspx) | Returns a reference to the first element in a vector. |
| [get\_allocator](https://msdn.microsoft.com/en-us/library/15efhsex.aspx) | Returns the **allocator** used by the vector. |
| [insert](https://msdn.microsoft.com/en-us/library/s5bta5ha.aspx) | Inserts an element or a number of elements into the vector at a specified position. |
| [max\_size](https://msdn.microsoft.com/en-us/library/k3k7ekc0.aspx) | Returns the maximum length of the vector. |
| [pop\_back](https://msdn.microsoft.com/en-us/library/59ykakh8.aspx) | Deletes the element at the end of the vector. |
| [push\_back](https://msdn.microsoft.com/en-us/library/7fthz5xd.aspx) | Adds an element to the end of the vector. |
| [rbegin](https://msdn.microsoft.com/en-us/library/eh974a8w.aspx) | Returns an iterator to the first element in a reversed vector. |
| [rend](https://msdn.microsoft.com/en-us/library/c1xese40.aspx) | Returns a reverse\_iterator that points to one before the first element. |
| [reserve](https://msdn.microsoft.com/en-us/library/f7yseh4d.aspx) | Reserves a minimum length of storage for a vector object. |
| [resize](https://msdn.microsoft.com/en-us/library/wezs0zy6.aspx) | Specifies a new size for a vector. |
| [shrink\_to\_fit](https://msdn.microsoft.com/en-us/library/dd647619.aspx) | Discards excess capacity. |
| [size](https://msdn.microsoft.com/en-us/library/3y41k4hb.aspx) | Returns the number of elements in the vector. |
| [swap](https://msdn.microsoft.com/en-us/library/8762zzx6.aspx) | Exchanges the elements of two vectors. |

[Operators](javascript:void(0))

|  |  |
| --- | --- |
| operator[] | Returns a reference to the vector element at a specified position. |
| [operator=](https://msdn.microsoft.com/en-us/library/dd673471.aspx) | Replaces the elements of the vector with a copy of another vector. |

Vectors are more powerful than arrays because the number of functions that are available for accessing and modifying vectors. Unfortunately, the [] operator still does not provide bounds checking. There is an alternative way of accessing the vector, using the function at, which does provide bounds checking at an additional cost. Let's take a look at several functions provided by the vector class:

unsigned int size(); Returns the number of elements in a vector

push\_back(*type* element); Adds an element to the end of a vector

bool empty(); Returns true if the vector is empty

void clear(); Erases all elements of the vector

*type* at(int n); Returns the element at index n, with bounds checking

also, there are several basic operators defined for the vector class:

= Assignment replaces a vector's contents with the contents of another

== An element by element comparison of two vectors

[] Random access to an element of a vector (usage is similar to that

of the operator with arrays.) Keep in mind that it does not provide

bounds checking.

Let's take a look at an example program using the vector class:

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main()**

**{**

**vector <int> example; //Vector to store integers**

**example.push\_back(3); //Add 3 onto the vector**

**example.push\_back(10); //Add 10 to the end**

**example.push\_back(33); //Add 33 to the end**

**for(int x=0; x<example.size(); x++)**

**{**

**cout<<example[x]<<" "; //Should output: 3 10 33**

**}**

**if(!example.empty()) //Checks if empty**

**example.clear(); //Clears vector**

**vector <int> another\_vector; //Creates another vector to store integers**

**another\_vector.push\_back(10); //Adds to end of vector**

**example.push\_back(10); //Same**

**if(example==another\_vector) //To show testing equality**

**{**

**example.push\_back(20);**

**}**

**for(int y=0; y<example.size(); y++)**

**{**

**cout<<example[y]<<" "; //Should output 10 20**

**}**

**return 0;**

**}**

The following example demonstrates various techniques involving a vector and C++ Standard Library algorithms, notably shuffling, sorting, finding the largest element, and erasing from a vector using the erase-remove idiom.

**#include <iostream>**

**#include <vector>**

**#include <array>**

**#include <algorithm> // sort, max\_element, random\_shuffle, remove\_if, lower\_bound**

**#include <functional> // greater**

**#include <iterator> //begin, end, cbegin, cend, distance**

**// used here for convenience, use judiciously in real programs.**

**using namespace std;**

**using namespace std::placeholders;**

**auto main(int, char\*\*)**

**-> int**

**{**

**std::array<int,4> arr{ 1, 2, 3, 4 };**

**// initialize a vector from an array**

**vector<int> numbers( cbegin(arr), cend(arr) );**

**// insert more numbers into the vector**

**numbers.push\_back(5);**

**numbers.push\_back(6);**

**numbers.push\_back(7);**

**numbers.push\_back(8);**

**// the vector currently holds { 1, 2, 3, 4, 5, 6, 7, 8 }**

**// randomly shuffle the elements**

**random\_shuffle( begin(numbers), end(numbers) );**

**// locate the largest element, O(n)**

**auto largest = max\_element( cbegin(numbers), cend(numbers) );**

**cout << "The largest number is " << \*largest << "\n";**

**cout << "It is located at index " << distance(largest, cbegin(numbers)) << "\n";**

**// sort the elements**

**sort( begin(numbers), end(numbers) );**

**// find the position of the number 5 in the vector**

**auto five = lower\_bound( cbegin(numbers), cend(numbers), 5 );**

**cout << "The number 5 is located at index " << distance(five, cbegin(numbers)) << "\n";**

**// erase all the elements greater than 4**

**numbers.erase( remove\_if(begin(numbers), end(numbers),**

**bind(greater<>{}, \_1, 4) ), end(numbers) );**

**// print all the remaining numbers**

**for(const auto& element : numbers)**

**cout << element << " ";**

**return 0;**

**}**

The output will be the following:

**The largest number is 8**

**It is located at index 6 (implementation-dependent)**

**The number 5 is located at index 4**

**1 2 3 4**