Vectors

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

<https://www.geeksforgeeks.org/vector-in-cpp-stl/>

1. [begin()](https://www.geeksforgeeks.org/vectorbegin-vectorend-c-stl/) – Returns an iterator pointing to the first element in the vector
2. [end()](https://www.geeksforgeeks.org/vectorbegin-vectorend-c-stl/) – Returns an iterator pointing to the theoretical element that follows the last element in the vector
3. [rbegin()](https://www.geeksforgeeks.org/vector-rbegin-and-rend-function-in-c-stl/) – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
4. [rend()](https://www.geeksforgeeks.org/vector-rbegin-and-rend-function-in-c-stl/) – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)
5. [cbegin()](https://www.geeksforgeeks.org/vector-cbegin-vector-cend-c-stl/) – Returns a constant iterator pointing to the first element in the vector.
6. [cend()](https://www.geeksforgeeks.org/vector-cbegin-vector-cend-c-stl/) – Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.
7. [crbegin()](https://www.geeksforgeeks.org/vectorcrend-vectorcrbegin-examples/) – Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
8. [crend()](https://www.geeksforgeeks.org/vectorcrend-vectorcrbegin-examples/) – Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

// C++ program to illustrate the

// iterators in vector

#include <iostream>

#include <vector>

using namespace std;

int main()

{

    vector<int> g1;

    for (int i = 1; i <= 5; i++)

        g1.push\_back(i);

    cout << "Output of begin and end: ";

    for (auto i = g1.begin(); i != g1.end(); ++i)

        cout << \*i << " ";

    cout << "\nOutput of cbegin and cend: ";

    for (auto i = g1.cbegin(); i != g1.cend(); ++i)

        cout << \*i << " ";

    cout << "\nOutput of rbegin and rend: ";

    for (auto ir = g1.rbegin(); ir != g1.rend(); ++ir)

        cout << \*ir << " ";

    cout << "\nOutput of crbegin and crend : ";

    for (auto ir = g1.crbegin(); ir != g1.crend(); ++ir)

        cout << \*ir << " ";

    return 0;

}

**Output:**

Output of begin and end: 1 2 3 4 5

Output of cbegin and cend: 1 2 3 4 5

Output of rbegin and rend: 5 4 3 2 1

Output of crbegin and crend : 5 4 3 2 1

1. [assign()](https://www.geeksforgeeks.org/vector-assign-in-c-stl/)– It assigns new value to the vector elements by replacing old ones
2. [push\_back()](https://www.geeksforgeeks.org/vectorpush_back-vectorpop_back-c-stl/) – It push the elements into a vector from the back
3. [pop\_back()](https://www.geeksforgeeks.org/vectorpush_back-vectorpop_back-c-stl/) – It is used to pop or remove elements from a vector from the back.
4. [insert()](https://www.geeksforgeeks.org/vector-insert-function-in-c-stl/) – It inserts new elements before the element at the specified position
5. [erase()](https://www.geeksforgeeks.org/vectorclear-vectorerase-c-stl/) – It is used to remove elements from a container from the specified position or range.
6. [swap()](https://www.geeksforgeeks.org/vectorat-vectorswap-c-stl/) – It is used to swap the contents of one vector with another vector of same type. Sizes may differ.
7. [clear()](https://www.geeksforgeeks.org/vectorclear-vectorerase-c-stl/) – It is used to remove all the elements of the vector container
8. [emplace()](https://www.geeksforgeeks.org/vector-emplace-function-in-c-stl/) – It extends the container by inserting new element at position
9. [emplace\_back()](https://www.geeksforgeeks.org/vectoremplace_back-c-stl/) – It is used to insert a new element into the vector container, the new element is added to the end of the vector

array of vectors

<https://www.geeksforgeeks.org/array-of-vectors-in-c-stl/>

An [**array**](https://www.geeksforgeeks.org/arrays-in-c-cpp/) is a collection of items stored at contiguous memory locations. It is to store multiple items of the same type together. This makes it easier to get access to the elements stored in it by the position of each element.

[**Vectors**](https://www.geeksforgeeks.org/vector-in-cpp-stl/) are known as [dynamic arrays](https://www.geeksforgeeks.org/how-do-dynamic-arrays-work/) with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container automatically.

Therefore, [**array**](https://www.geeksforgeeks.org/introduction-to-arrays/)**of**[**vectors**](https://www.geeksforgeeks.org/vector-in-cpp-stl/) is two dimensional array with fixed number of rows where each row is vector of variable length. Each index of array stores a vector which can be traversed and accessed using [iterators](https://www.geeksforgeeks.org/iterators-c-stl/).

**Syntax:**

vector <data\_type> V[size];

**Example:**

vector <int> A[5];

where A is the array of vectors of int of size 5

**For Example:**

for i in [0, n) {

A[i].push\_back(35)

}

Above pseudo-code inserts element 35 at every index of **vector <int> A[n]**.

**Traversal:** Traversal in an array of vectors is perform using [iterators](https://www.geeksforgeeks.org/iterators-c-stl/).

**For Example:**

for i in [0, n) {

for(iterator it = A[i].begin();

it!=A[i].end(); it++) {

print(\*it)

}

}

Above pseudo-code traverses **vector <int> A[n]** at each index using starting iterators **A[i].begin()** and ending iterator **A[i].end()**. For accessing the element it uses **(\*it)** as iterators are pointers pointing to elements in **vector <int> A[n]**.

Operators: <http://www.cplusplus.com/reference/vector/vector/operators/>

Sorting a vector : <https://www.gormanalysis.com/blog/sorting-a-vector-in-cpp/>

#include <iostream> // std::cout

#include <algorithm> // std::sort

#include <vector> // std::vector

int main()

{

std::vector<int> foo = {5, 3, 10, 1, 7}; // Make values

std::sort(foo.begin(), foo.end());

// Sort

for(int &x : foo) std::cout << x << " "; // Print

}