**C++ Vectors**

**A. What are vectors?**

Vectors are sequence containers that fall under the category of class templates. They are used to store information in a linear fashion. Elements of a single data type can be stored in a vector. In a vector, we can access elements faster in comparison to arrays. Although, insertion and deletion at some random position, except at the end is comparatively slower. And inserting any element at the end happens faster.

**B. Using a vector in our programs**

To be able to use vectors in our code, the header file <vector> must be included. And the syntax to define a vector is

vector<data\_type> vector\_name;

The data\_type could be replaced by any data type. One benefit of using vectors is that we can insert as many elements as we want in a vector, without having to put some size parameter as we do in an array.

Vectors provide certain methods to be used to access and utilize the elements of a vector, the first one being, the push\_back method. To access all the methods and member functions in detail, one can visit this site, [std::vector - C++ Reference](https://www.cplusplus.com/reference/vector/vector/).

**C. Initialising a vector**

A vector could be initialized in different ways:

* In the first method, a vector could be initialized with all the elements inserted in the vector at the time it is defined.

vector<int> v = {1, 2, 3, 4};

* Another method to initialize a vector is where we pass two parameters along with its definition where the first parameter defines the size of the vector and the second parameter is the uniform value it will have at all its positions.

vector<int> v(4, 10);

Here, v is of size 4 with value 10 at all positions.

**D. Inserting elements in a vector**

We can insert elements in a vector in different ways. The most economical way to insert an element in a vector is when we insert it at the rear end using the push\_back method. The push\_back() function gets the elements to be inserted as a parameter and the element gets pushed at the back.

#include <iostream>

#include <vector>

using namespace std;

void display(vector<int> v)

{

cout << "The elements are: ";

for (auto it : v)

{

cout << it << " ";

}

cout << endl;

}

int main()

{

vector<int> v = {1, 2, 3, 4};

display(v);

v.push\_back(5); //5 is inserted at the back

display(v);

}

Output

The elements are: 1 2 3 4

The elements are: 1 2 3 4 5

Another method to insert an element allows us to insert at any random position. Here, we use the insert() method. The syntax for using the insert method is

vector\_name.insert(iterator, element);

Here, the iterator is the pointer to that position where the element gets inserted. An example that demonstrates the use of the insert() method is

#include <iostream>

#include <vector>

using namespace std;

void display(vector<int> v)

{

cout << "The elements are: ";

for (auto it : v)

{

cout << it << " ";

}

cout << endl;

}

int main()

{

vector<int> v = {1, 2, 3, 4};

display(v);

v.insert(v.begin(), 5);

display(v);

}

Output

The elements are: 1 2 3 4

The elements are: 5 1 2 3 4

Here, we used the begin() method which returned an iterator to the first position of the vector. We can manipulate it accordingly to point at the desired position by general arithmetic operations.

**E. Accessing/Changing elements in a vector**

Changing any element at any position is similar to accessing them. Any element at any specified position could be accessed using its index number as it was done for arrays.

vector<int> v = {1, 2, 3, 4};

cout << "Element at index 0 is " << v[0] << endl;

Copy

Output

Element at index 0 is 1

Another method that gets added for vectors is the at() method which accepts the index number as its parameter and returns the element at that position.

vector<int> v = {1, 2, 3, 4};

cout << "Element at index 2 is " << v.at(2) << endl;

Output

Element at index 2 is 3

**F. Removing elements from a vector**

We can remove elements from a vector in different ways. The most economical way to remove an element from a vector is when we remove it from the rear end using the pop\_back method. The pop\_back() function needs nothing as a parameter and the element gets popped from the back.

#include <iostream>

#include <vector>

using namespace std;

void display(vector<int> v)

{

cout << "The elements are: ";

for (auto it : v)

{

cout << it << " ";

}

cout << endl;

}

int main()

{

vector<int> v = {1, 2, 3, 4};

display(v);

v.pop\_back(); //4 gets popped from the back

display(v);

}

Output

The elements are: 1 2 3 4

The elements are: 1 2 3

Another method to remove an element allows us to remove it from any random position. Here, we use the erase() method. The syntax for using the erase method is

vector\_name.erase(iterator);

Here, the iterator is the pointer to that position where the element gets popped from. An example that demonstrates the use of the erase() method is