

**EC-204 Data Structures and Object Oriented**

**DE-41 Mechatronics**

Lab Report #3

**Name :Muhammad Hamza Shah**

**Reg# 325390**

**Syndicate: A**

Submitted to: Dr Waqar Shahid/LE Arshia Arif

**ABSTRACT:**

Vectors in C ++ are sequential containers that represent configurations that can change in size. They use the final combinations of their objects, which means that their objects can be found and used indexes in standard directions on its objects, and in the same way as in arrays. The following Points will be included in this report:

1. [What is Vector in C++?](https://www.edureka.co/blog/vectors-in-cpp/#what) 
   1. [Is vector Vectors in C++ are sequence containers representing arrays that can change in size. They use contiguous storage locations for ordered in C++?](https://www.edureka.co/blog/vectors-in-cpp/#ordered)
   2. [How are vectors stored in C++?](https://www.edureka.co/blog/vectors-in-cpp/#store)
2. [Member functions of Vectors](https://www.edureka.co/blog/vectors-in-cpp/#member) 
   1. [Modifiers](https://www.edureka.co/blog/vectors-in-cpp/#modifiers)
   2. [Iterators](https://www.edureka.co/blog/vectors-in-cpp/#iterators)

**INTRODUCTION:**

Vectors in C ++ are powerful arrays used to store data. Unlike the same members, which are used to store consistent and naturally occurring data, Vectors provide more flexibility to the system. Vectors can grow on their own if an object is inserted or removed depending on the need for the work to be done. It is not the same for the same members where only the value of a given number under one variable name can be set.

**Procedure/Elaboration:**

Vector items are stored in a nearby store for retrieval and crossing using iterators. In vectors, data is entered at its end. Entering an object in the end takes time to vary, as sometimes there may be a need to extend the vector. Deleting the last item takes permanent time because no size change takes place.

**Syntax:**

|  |  |
| --- | --- |
| 1 | vector< object\_type > variable\_name; |

Iterators:

1. **begin():** This function returns an iterator pointing to the **first** element in the vector.
2. **end():** The end() function returns an iterator pointing to the **last** element in the vector.

**Example:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**int main()**

**{**

**vector<int> vec1;**

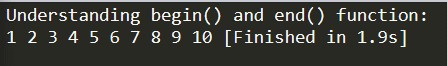
**for (int i = 1; i <= 10; i++) vec1.push\_back(i);**

**cout << "Understanding begin() and end() function: " << endl; for (auto i = vec1.begin(); i != vec1.end(); ++i) cout << \*i << " ";**

**return 0;**

**}**

**Output:**



In the above example, we can see the use of begin() and end() function. First, we define a vector vec1, we push back values in it from 1 to 10 using a for a loop. Then we print the values of our vectors using for loop, we use begin( ) and end( ) function to specify the start and endpoint of our for a loop.

Modifiers:

1. **push\_back():** The function pushes the elements into a vector from the back. If the type of object passed as a parameter in the push\_back() is not same as that of the vector an exception is thrown.
2. **assign():** It assigns a new value to the vector elements by replacing old ones.
3. **pop\_back():** The pop\_back() function is used to pop or remove elements from a vector from the back. It reduces the size of the vector by one element.
4. **insert():** This function inserts new elements before the element before the position pointed by the iterator. We can also pass a third argument count, that counts the number of times the element is to be inserted before the pointed position.
5. **erase():** erase() function is used to remove elements from a container from the sp
6. **ecified position or range**. We can either pass the position of the specific elements needs need to remove or we can pass the starting point and endpoint of a range of elements.
7. **swap():** swap() function is used to swap the contents of one vector with another vector of the same type. Sizes may differ.
8. **clear():** clear() function is used to remove all the elements of the vector container

**Example:**

/ Modifiers in vector

//#include <bits/stdc++.h>

#include<iostream> #include <vector> using namespace std;

int main()

{

// Assign vector vector<int> vec; // this is a container/LIST of type int

// fill the array with 12 seven times vec.assign(7, 12); // modifier functions of a vector

cout << "The vector elements are: "<<endl; for (unsigned long long int i = 0; i <vec.size(); i++) cout << vec[i] << " "<<endl;

// inserts 24 to the last position vec.push\_back(24);

int s = vec.size(); cout << "The last element is: " << vec[s - 1]<<endl; cout<<vec[s]<<endl;

// prints the vector cout << "The vector elements after push back are: "<<endl; for (unsigned long long int i = 0; i <vec.size(); i++) cout << vec[i] << " "<<endl;

// removes last element vec.pop\_back(); // removes the last element

// prints the vector cout << "The vector elements after pop\_back are: "<<endl; for (unsigned long long int i = 0; i < vec.size(); i++) cout << vec[i] << " "<<endl;

// inserts 10 at the beginning vec.insert(vec.begin(), 10); // is function that insert an element at a specific place

cout << "The first element after insert command is: " << vec[0]<<endl;

// removes the first element

vec.erase(vec.begin());

cout << "The first element after erase command is: " << vec[0]<<endl;

// inserts at the beginning

vec.emplace(vec.begin(), 5); // you might not need to LEARN BY HEART these function cout << "The first element emplace is: " << vec[0]<<endl;

// Inserts 20 at the end vec.emplace\_back(20); s = vec.size(); cout << "The last element after emplace\_back is: " << vec[s - 1]<<endl;

// erases the vector vec.clear(); // deletes all the memory elements in the vector cout << "Vector size after clear(): " << vec.size()<<endl;

// two vector to perform swap vector<int> obj1, obj2;

//first vector obj1.push\_back(2); obj1.push\_back(4); // second vector obj2.push\_back(6); obj2.push\_back(8);

cout << "Vector 1: "; for (unsigned long long int i = 0; i < obj1.size(); i++) cout << obj1[i] << " "<<endl;

cout << "Vector 2: "; for (unsigned long long int i = 0; i < obj2.size(); i++) cout << obj2[i] << " "<<endl;

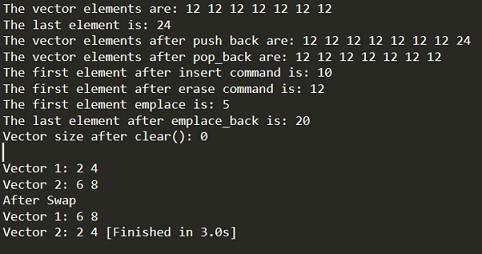
// Swaps obj1 and obj2 obj1.swap(obj2);

cout << "After Swap nVector 1: "; for (unsigned long long int i = 0; i < obj1.size(); i++) cout << obj1[i] << " "<<endl;

cout << "Vector 2: "; for (unsigned long long int i = 0; i < obj2.size(); i++) cout << obj2[i] << " "<<endl;

}

**Output:**



**CONCLUSION:**

In this lab we got to know the different member functions of Vectors and the working of each one of them. We got an idea of what exactly is a namespace in C++ and what are the different ways of defining it.