Vector

<Jaydeep Shah>

Disadvantage of Array? : We can not change the size of Array during runtime.
In C++, the Vector data structure is preferred over Arrays due to its **Dynamic memory allocation**. Unlike Arrays, **Vectors allow us to change their size during runtime, providing flexibility**. Additionally, Vectors come with built-in functions that are helpful for tasks such as inserting and sorting data efficiently.

Note:**Vector library file is required for using vector datatype.**

Code:

```
#include<iostream>
#include <bits/stdc++.h>
#include<vector>                    //Vector library required


using namespace std;


int main()
{
        //---------------------------------------------------------------------
        //Creating vector with "int" datatype and name "v".
        vector<int>  v;  //No memory (byte) allocation during creation
        cout << "Size of vector v  = "<<v.size() << endl;
        //---------------------------------------------------------------------


        //---------------------------------------------------------------------
        //insert elemnts during runtime
        v.push_back(1);             //Element with value "1" push at the end of vector
        v.push_back(2);             //Element with value "2" push at the end of vector
        v.push_back(3);             //Element with value "3" push at the end of vector
        //---------------------------------------------------------------------


        //---------------------------------------------------------------------
        //print size after new elements
        cout << "Size of integer = " << sizeof(int) << endl;
        cout << "New Size of vector v (In terms of element)  = "<<v.size() << endl;
       //Here siz means number of elemnts
        //---------------------------------------------------------------------

        //---------------------------------------------------------------------
        //print vector
        for(int a =0; a < v.size(); a++)
        {
```

```
		//Access of element just like array
		cout << v[a] <<endl;		//Print vector individual element value
	}
	//---------------------------------------------------------------------

	//---------------------------------------------------------------------
	//Another way of accessing elements using itterator
	//An iterator is an object (like a pointer) that points to an element inside the container. We can use iterators to move through the contents of the container
	cout <<endl <<"Access using itterator " << endl;
	vector<int> :: iterator itr;    //create iiterator  (Pointer to point element of vector type)
	for(itr = v.begin(); itr != v.end(); itr++)
	{
		cout << *itr << endl;		//Acess using itterator
	}
	//---------------------------------------------------------------------

	//---------------------------------------------------------------------
	//Remove elemets for vector
	v.pop_back();			//Last element and it's memory now removed from vector v
	cout << endl << "After popping laste lement from vector new data = " << endl;
	for(itr = v.begin(); itr != v.end(); itr++)
	{
		cout << *itr << endl;		//Acess using itterator
	}
	//---------------------------------------------------------------------

	//---------------------------------------------------------------------
	//Swap two vector elements
	//Syntax vector_name(size,value)
	vector<int> v2(3,12);   //Vector v2 with size=3 and all element with value 12
	cout<<endl <<"Vector with size 3 and value 12 of all elements"<<endl;
	vector<int> :: iterator itr2;
	for(itr2 = v2.begin();itr2 != v2.end(); itr2++)
	{
		cout << *itr2 << endl;
	}

	//Swap
	swap(v,v2);
	//After swaping print both vector
	cout<<endl <<"After swapping V1: "<<endl;
	for(itr2 = v.begin();itr2 != v.end(); itr2++)
	{
		cout << *itr2 << endl;
	}

	 cout<<endl <<"After swapping V2: "<<endl;
	for(itr2 = v2.begin();itr2 != v2.end(); itr2++)
	{
		cout << *itr2 << endl;
	}
```

```
	//-------------------------------------------------------------------

	//-------------------------------------------------------------------
	//Change the elements of vector
	//This way you can only allocate value whoes memory already allocated, v[3] = x  will not work here
	v[0] = 15;
	v[1] = 11;
	v[2] = 13;

	//Ascending order sorting
	sort(v.begin(),v.end());
	cout<<endl <<"After sorting in v2: "<<endl;
	for(itr2 = v.begin();itr2 != v.end(); itr2++)
	{
		cout << *itr2 << endl;
	}
	//-------------------------------------------------------------------
	return 0;
}
```

OUTPUT:

Some other useful function:

1) erase() - To remove particular element / range of elements from vector container.

Example : v.erase(3) //Delete element whoese position is #3.

Example : v.erase(start_pos,end_pos) // Deletion inrange

NOTE: Because of dynamic memory allocation used in vector concept, need to remove allocated memory after complition of usage of vector. Otherwise lead to Heap flow during creating any of new vector.

v.clear() = Remove all elements from vector container

Example:

```
v.clear();
if(v.empty())
{
   cout << " \n Vector is empty ";
}
```

Example 2:

```
while(!v.empty())
{
   v.pop_back();
}
if(v.empty())
{
   cout << " \n Vector is empty ";
}
```