Vectors

A vector is a sequence container class that implements dynamic array, means size automatically changes when appending elements. A vector stores the elements in contiguous memory locations and allocates the memory as needed at run time.

Difference between vector and array

An array follows static approach, means its size cannot be changed during run time while vector implements dynamic array means it automatically resizes itself when appending elements.

Syntax:

Consider a vector 'v1'. Syntax would be:

vector<object_type> v1;

Example:

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
vector<string> v1;
v1.push_back("HiHello");
v1.push_back("WELCOME");
for(vector<string>::iterator itr=v1.begin();itr!=v1.end();++itr)
cout<<*itr;
return 0;   }
```

Output:

HiHello WELCOME

Functions of Vector

Function	Description
at()	It provides a reference to an element.
back()	It gives a reference to the last element.
front()	It gives a reference to the first element.
swap()	It exchanges the elements between two vectors.
push_back()	It adds a new element at the end.
pop_back()	It removes a last element from the vector.
empty()	It determines whether the vector is empty or not.
insert()	It inserts new element at the specified position.
erase()	It deletes the specified element.
resize()	It modifies the size of the vector.
clear()	It removes all the elements from the vector.

size()	It determines a number of elements in the vector.
capacity()	It determines the current capacity of the vector.
assign()	It assigns new values to the vector.
operator=()	It assigns new values to the vector container.
operator[]()	It access a specified element.
end()	It refers to the past-lats-element in the vector.
emplace()	It inserts a new element just before the position pos.
emplace_back()	It inserts a new element at the end.
rend()	It points the element preceding the first element of the vector.
rbegin()	It points the last element of the vector.
begin()	It points the first element of the vector.
max_size()	It determines the maximum size that vector can hold.
cend()	It refers to the past-last-element in the vector.
cbegin()	It refers to the first element of the vector.
crbegin()	It refers to the last character of the vector.
crend()	It refers to the element preceding the first element of the vector.
data()	It writes the data of the vector into an array.
shrink_to_fit()	It reduces the capacity and makes it equal to the size of the vector.

Example Programs of vector:

1. 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
```

2. C++ program to illustrate the capacity function 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 << "Size : " << g1.size();
	cout << "\nCapacity : " << g1.capacity();
	cout << "\nMax_Size : " << g1.max_size();

	// resizes the vector size to 4
	g1.resize(4);

	// prints the vector size after resize()
	cout << "\nSize : " << g1.size();

	// checks if the vector is empty or not
	if (g1.empty() == false)
		cout << "\nVector is not empty";
	else
		cout << "\nVector is empty";

	// Shrinks the vector
	g1.shrink_to_fit();
	cout << "\nVector elements are: ";
	for (auto it = g1.begin(); it != g1.end(); it++)
		cout << *it << " ";

	return 0;
}
```

Output:
Size : 5
Capacity : 8
Max_Size : 4611686018427387903
Size : 4
Vector is not empty
Vector elements are: 1 2 3 4

3. // C++ program to illustrate the element access in vector

```
#include <bits/stdc++.h>
using namespace std;
```

```
int main()
{
	vector<int> g1;

	for (int i = 1; i <= 10; i++)
		g1.push_back(i * 10);

	cout << "\nReference operator [g] : g1[2] = " << g1[2];

	cout << "\nat : g1.at(4) = " << g1.at(4);

	cout << "\nfront() : g1.front() = " << g1.front();

	cout << "\nback() : g1.back() = " << g1.back();

	// pointer to the first element
	int* pos = g1.data();

	cout << "\nThe first element is " << *pos;
	return 0;
}
```

Output:

```
Reference operator [g] : g1[2] = 30
at : g1.at(4) = 50
front() : g1.front() = 10
back() : g1.back() = 100
The first element is 10
```

4. C++ program to illustrate the Modifiers in vector

```
#include <bits/stdc++.h>
#include <vector>
using namespace std;

int main()
{
	// Assign vector
	vector<int> v;
```

```
// fill the vector with 10 five times
v.assign(5, 10);

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

// inserts 15 to the last position
v.push_back(15);
int n = v.size();
cout << "\nThe last element is: " << v[n - 1];

// removes last element
v.pop_back();

// prints the vector
cout << "\nThe vector elements are: ";
for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";

// inserts 5 at the beginning
v.insert(v.begin(), 5);

cout << "\nThe first element is: " << v[0];

// removes the first element
v.erase(v.begin());

cout << "\nThe first element is: " << v[0];

// inserts at the beginning
v.emplace(v.begin(), 5);
cout << "\nThe first element is: " << v[0];

// Inserts 20 at the end
v.emplace_back(20);
```

```
	n = v.size();
	cout << "\nThe last element is: " << v[n - 1];

	// erases the vector
	v.clear();
	cout << "\nVector size after clear(): " << v.size();

	// two vector to perform swap
	vector<int> v1, v2;
	v1.push_back(1);
	v1.push_back(2);
	v2.push_back(3);
	v2.push_back(4);

	cout << "\n\nVector 1: ";
	for (int i = 0; i < v1.size(); i++)
		cout << v1[i] << " ";

	cout << "\nVector 2: ";
	for (int i = 0; i < v2.size(); i++)
		cout << v2[i] << " ";

	// Swaps v1 and v2
	v1.swap(v2);

	cout << "\nAfter Swap \nVector 1: ";
	for (int i = 0; i < v1.size(); i++)
		cout << v1[i] << " ";

	cout << "\nVector 2: ";
	for (int i = 0; i < v2.size(); i++)
		cout << v2[i] << " ";
	return 0;
}
```

Output:

```
The vector elements are: 10 10 10 10 10
The last element is: 15
The vector elements are: 10 10 10 10 10
The first element is: 5
The first element is: 10
The first element is: 5
The last element is: 20
Vector size after erase(): 0

Vector 1: 1 2
Vector 2: 3 4
After Swap
Vector 1: 3 4
Vector 2: 1 2
```