Vectors

11 June 2024 02:11 PM

A Vector is a Class template.

↓

A Blueprint from which specific class types can be created.

Template :

- Not Functions or classes
- Guidelines for the complier to use to create classes or functions.

For Vectors:

We Specify the Type of the objects the vector will hold

Syntax --- vector<int> myvec

Vector is a dynamic for memory size

Default vector is vector<T> v1 which will create an empty vector to v1

Vector Copying :

vector<T> v2 (v1) vector<T> v2 = v1

v2 has the copy of each element of v1

Vector values can be copied into only to the same datatype of vector

List Initialising :

vector<T> v1 = {a,b,c,d,.....} vector<T> v1{a,b,c,d,.....}

A Vector needed to enclosed in curly braces only

Creating specific number of elements :

vector <T> v1(n,val) vector <T> v1(n)

Example:

```
vector<int> v1(5); //v1 has 5 elements with value 0
vector<int> v2{5}; //v2 has 1 elements with value 5
vector<int> v3(5,2); //v3 has 5 elements with value 2
vector<int> v4{5,2}; //v4 has 2 elements with value 5,2
vector<string> v5{"cup"}; //v5 has 1 element string cup
vector<string> v6("cup"); //error
```

Adding Elements to a Vector:

The push_back operation

Takes a value and "pushes" that value as a new last element onto the "back" of the vector.

Vector Basic Operations:

v.empty()	Returns • True, if v is empty • False, if v is not empty
v.size()	Returns the number of elements in v.
v.push_back(t)	Adds an element with value t to the end of v.
v[n]	Returns a reference to the element at the position n in v.
v1 = v2	Replaces the elements in v1 with a copy of the elements in v2.
v1 = {a,b,c,d,.....}	Replaces the elements in v1 with a copy of the elements in the comma-separated list.
v1 == v2	v1 and v2 are R equal if they have the same number of elements and each element in v1 is equal to the corresponding element in v2.
v1 ! = v2	
v1 < v2	
v1 > v2	
v1 < = v2	
v1 > = v2	

Simple exam of Vector

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

int main() {
    vector<int> myvec;
    myvec ={1,2,3,4,5};
    for(int i:myvec)
      cout<<i<<endl;
return 0;
}
```

Output

```
1
2
3
4
5
```

Vector is a sequential container to store elements and not index based. Array stores a fixed-size sequential collection of elements of the same type and it is index based. Vector is dynamic in nature so, size increases with insertion of elements.

Copy Vector

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> myvec;
    myvec ={1,2,3,4,5};
    vector<int> vec(myvec);
    for(int i:vec)
      cout<<i<<endl;
return 0;
}
```

Output

```
1
2
3
4
5
```

Specific elements Initialising

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> myvec(5,-2);
    vector<string> svec(4,"coding");
    vector<int> ivec(6);
    for(int i:myvec)
      cout<<i<<" ";
    cout<<endl;
    for(string s:svec)
      cout<<s<<" ";
    cout<<endl;
    for(int j : ivec)
      cout<<j<<" ";
return 0;
}
```

```
-2 -2 -2 -2 -2
coding coding coding coding
0 0 0 0 0 0
```

Adding Elements to a vector using push_back

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
  vector<int> myvec;
  for(int i=0;i<=15;i++){
    myvec.push_back(i);
  }
  for(int i: myvec)
    cout<<i<<" ";
return 0;
}
```

```
0 1 2 3 4 5 6 7 8 9 10
```

Adding Elements in vector of String

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
  string word;
  vector<string> mytext;
  while(cin>>word){
    mytext.push_back(word);
  }
  for(string i:mytext)
     cout<<i<<endl;
return 0;
}
```

```
Hey! How's your coding experience?^Z
^Z
Hey!
How's
your
coding
experience?
```

Vector Operations

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
  vector<int> myvec;
  cout<<"is myvec empty? "<<myvec.empty()<<endl;
  myvec={1,2,3,4,5};
  cout<<"is myvec empty? "<<myvec.empty()<<endl;
  cout<<"Size of myvec = "<<myvec.size()<<endl;
  myvec.push_back(6);
  for(int i:myvec)
    cout<<i<<" ";
  cout<<endl<<"The third element in myvec = "<<myvec[2];
  vector<int> myvec2={7,8,9,10};
  myvec = myvec2;
  cout<<endl<<"New values in myvec = ";
  for(int i:myvec)
    cout<<i<<" ";
  myvec={11,12,13,14,15};
  cout<<endl<<"Replaced values of myvec = ";
  for(int i:myvec)
    cout<<i<<" ";
  cout<<endl;
  if(myvec == myvec2){
    cout<<"The two vectors are equal"<<endl;
  }
  else
    cout<<"The two vectors are not equal"<<endl;
return 0;
}
```

```
is myvec empty? 1
is myvec empty? 0
Size of myvec = 5
1 2 3 4 5 6
The third element in myvec = 3
New values in myvec = 7 8 9 10
Replaced values of myvec = 11 12 13 14 15
The two vectors are not equal
```