What is a vector?
vector is one of the most powerful standard template library data structures. Vector is a similar Data Structure like array but the difference is the array is static and vector is dynamic. Why the array is static and vector is dynamic will cover in the next lesson. But for today's lesson I can only tell you that when we declare an array we have to declare the size of the array this is because it is static but we don't need to declare the size of the vector because it is dynamic.Vector dynamically increases its size according to its need.

How can we declare a vector?
Let's see how we can declare an array. We can declare an array like this :

```
int main()
{
   int a[20];
   // data_type name_of_array[size_of_array];
}
```

First we define the datatype of all elements of an array, then we define the name of the array and then in a square bracket we define the size. Declaration of vectors is almost similar. We declare a vector like this :

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

int main()
{
   vector <int> v;
   // vector<data_type> name_of_vector;
}
```

Vector is included in the vector header file so we have to include it first and we need to use namespace std to declare vectors like above code. Otherwise we have to declare vector like this:

```
#include <vector>
int main()
{
   std::vector <int> v;
   // std:: as we didn’t used using namespace std;
}
```

First of all we tell our program that we are declaring a vector then in an angle bracket we define the datatype of the elements in the vector. Then we define the name of the vector but look we don't need to define the size of the vector as the vector is dynamic it will increase its size dynamically according to its need.

How to append elements in a vector?

In an array we can assign elements in any index as the array is static and we have defined it's size so it makes available all indexes for us. But vector is dynamic and we didn't define the size of it so we have to append elements one by one in a vector and for this it has a function **push_back()** using push back we can insert elements in it for example :

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

int main()
{
   vector <int> v; // empty vector
   v.push_back( 4 ); // v = {4} 4 is assigned in 0-no index
   v.push_back( 3 ); // v = {4, 3} 3 is assigned in 1-no index
   v.push_back( 5 ); // v = {4, 3, 5} 5 is assigned in 2-no
index
}
```

Look at the code carefully we declared a vector. The vector is initially empty but when we append 4 in the vector using push_back() function, 4 is assigned in the 0-no index of the vector and the vector becomes like this **v = {4}.**
When we append 3 in the vector using push_back() function, 3 is assigned in the 1-no index of the vector and the vector becomes like this **v = {4, 3}.**
When we append 5 in the vector using push_back() function, 5 is assigned in the 2-no index of the vector and the vector becomes like this **v = {4, 3, 5}.**

This is how we can assign or append elements in a vector.

How to access elements in a vector?

We can access element of a vector same as array for example:

```
#include <iostream>
```

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

int main()
{
   vector <int> v; // empty vector
   v.push_back( 4 ); // v = {4} 4 is assigned in 0-no index
   v.push_back( 3 ); // v = {4, 3} 3 is assigned in 1-no index
   v.push_back( 5 ); // v = {4, 3, 5} 5 is assigned in 2-no index

   cout << v[2] << endl; // prints 2 no index value 5.
}
```

Look, we can print the 2-no index element the same as array. We can access any index i like v[i], but i should be less than the size of the vector. We can also assign data in available indexes. For example :

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
  vector <int> v; // empty vector
  v.push_back( 4 ); // v = {4} 4 is assigned in 0-no index
  v.push_back( 3 ); // v = {4, 3} 3 is assigned in 1-no index
  v.push_back( 5 ); // v = {4, 3, 5} 5 is assigned in 2-no index
  v[1] = 10;
  cout << v[1] << endl; // prints 1 no index value 10.
}
```

Look initially the 1-no indexed value was 3. We changed it to 10 by assigning 10 in 1 no index.

How can we get the size of the vector?

There is a built-in function size() by which we can get the size of any vector for example

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

```
int main()
{
   vector <int> v; // empty vector
   v.push_back( 4 ); // v = {4} 4 is assigned in 0-no index
   v.push_back( 3 ); // v = {4, 3} 3 is assigned in 1-no index
   v.push_back( 5 ); // v = {4, 3, 5} 5 is assigned in 2-no index
   v.push_back( 10 ); // v = {4, 3, 5, 10} 10 is assigned in 3-no index
   v.push_back( 30 ); // v = {4, 3, 5, 10, 30} 30 is assigned in 4-no
index

   cout << v.size() << endl;// prints 5 the size of the vector
}
```

Look in the above vector where 5 elements are pushed. We can print the size or get the size by **v.size().** Now let's print all elements of a vector:

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

int main()
{
   vector <int> v; // empty vector
   v.push_back( 4 ); // v = {4} 4 is assigned in 0-no index
   v.push_back( 3 ); // v = {4, 3} 3 is assigned in 1-no index
   v.push_back( 5 ); // v = {4, 3, 5} 5 is assigned in 2-no index
   v.push_back( 10 ); // v = {4, 3, 5, 10} 10 is assigned in 3-no index
   v.push_back( 30 ); // v = {4, 3, 5, 10, 30} 30 is assigned in 4-no
index

   //Print all elements of a vector

   for ( int i = 0; i < v.size(); i++ ) {
       cout << v[i] << " ";
   }

   cout << endl;
```

```
    //Output : 4 3 5 10 30
}
```

So in this lesson we have learnt

- What is a vector
- How to declare a vector
- How to append elements in vector
- How to assign any data in available index of a vector
- How to access any index of a vector
- How to get the size of a vector
- How to print all the elements of a vector

That’s all for today. Hope you enjoyed today's lesson. See you in the next lesson.