Container is the idea of data structures. You will learn some of them in 131 class. Basically definition is how you store the data. Since you all know array now, I’m going to introduce one pre-defined library that provide the container very similar to array, which is called “**vector**”. You will like to use vector instead array because it is so convenience!

Firstly you can check here for more information:

[http://www.cplusplus.com/reference/vector/vector/](http://www.cplusplus.com/reference/vector/vector/%20)

For your convenience, a quick definition of vector is:

Vector is an dynamic array which can increase the size(double the size) automatically if the capacity is not enough. So you can keep adding elements in a vector without worrying about overload. There are some very easy to use functions provided by the <vector> library. You can also check the detail online. Usually you must know is clear, push\_back, pop\_back, insert, size ….

Let’s see the example:

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

vector<int> V;

for (int i = 0; i < 9; i++)

{

V.push\_back(i \* 3);

}

cout << "The size is: " << V.size() << endl;

V.pop\_back();

cout << "The size after pop\_back() is: " << V.size() << "\n" << endl;

cout << "print the vector by iterator:" << endl;

vector<int>::iterator iter = V.begin();

while (iter != V.end())

{

cout << \*iter << " ";

iter++;

}

cout << "\n" << endl;

cout << "print the fourth element by iterator:" << endl;

iter = V.begin() + 3;

cout << \*iter << "\n" << endl;

cout << "print the vector by index:" << endl;

int n = V.size(); //now n = 8 (the size of the vector)

for (int i = 0; i < n; i++)

{

cout << V[i] << " ";

}

cout << "\n" << endl;

cout << "print the fourth element by index:" << endl;

cout << V[3] << "\n" << endl;

cout << "print The vector after random shuffle:" << endl;

random\_shuffle(V.begin(), V.end());  // now let's make this vector become a  random list

for (int i = 0; i < n; i++)

{

cout << V[i] << " ";

}

cout << "\n" << endl;

cout << "print the vector after sorting it;" << endl;

sort(V.begin(), V.end()); // this is how you sort a vector by <algorithm> library

for (int i = 0; i < n; i++)

{

cout << V[i] << " ";

}

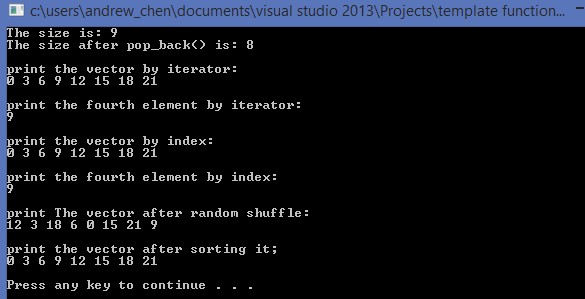
cout << "\n" << endl;

system("pause");

return 0;

}

Output:



Remember you can still use index to access data in the vector just like how you do with array. But if you want to insert value in the vector, firstly use push\_back function or insert function. After you insert a value in the vector, then you can use the index to access the location ( modify the value)

For example, if you define a integer vector:

**vector<int> V;**

now you want to insert a value 10 in the first location of this vector. You can’t directly do this:

**V[0] = 10;**

It won’t work because you haven’t create(reserve) the space for the vector (size still 0). So you have to use push\_back first:

**V.push\_back(10);**

Now your vector become a size 1 vector (dynamic array), so you can use index V[0] to access the location:

**V[0] = 999;**

**cout << V[0];**

Those will work now.

By the way, you have already learn this function “ramdom\_shuffle” which can re-arrange the contents of a vector by this way:

Assume the name of the vector is v.

**Random\_shuffle(v.begin(), v.end());**

But how about use random\_shuffle with a normal array ?

Int a[9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

You just need to put the address of the element of this array. Like this:

**random\_shuffle(&a[0], &a[8]);**

**This Wednesday will be our last SI session this semester, I plan to show you how to use stop point tool in visual studio for debug! It will be very useful if you prefer to use visual studio.**

**The same time : 1:00 – 2:15 pm**

**The same location: ULC (second floor of library)**

**Hope to see you. Good luck for final !**