National Chiao Tung University
Department of Electrical and Computer Engineering
Computer Intelligence on Automation(C.I.A.) Lab

Laboratory Manual 13
June 11, 2012
Prof. Hung-Pin(Charles) Wen

UEE1303(1070) S12: Object-Oriented Programming
Standard Libraries on Containers and Algorithms

What you will learn from Lab 13

In this laboratory, you will learn how to use STL containers and generic algorithms provided by standard library

TASK 13-1 VECTOR

✓   A container is an object whose main purpose is to hold other objects. A vector contains an array of n objects indexed from 0 to n-1.

```cpp
// lab13-1-1.cpp
#include <iostream>
#include <vector>
using std::cout;     using std::endl;
using std::vector;

int main()
{
    int n = 10;
    vector<int> vec1(n);                 // allocate a vector with 10 elements

    for (int i=0;i<vec1.size();i++)
        vec1[i] = i * i;                 // use subscripting to access elements

    for (int i=0;i<vec1.size();i++)
        cout << vec1[i] << " ";
    cout << endl;

    vector<int> vec2;                     // allocate an empty vector
    for (int i=0;i<n;i++)
        vec2.push_back(i * 2);           // use push_back() to add elements

    vector<int>::const_iterator iter;
    for (iter = vec2.begin(); iter!= vec2.end(); iter++)
        cout << iter << " ";             // use iterator to traverse container
    cout << endl;

    return 0;
}
```

➢   Please fix the compiler error here.

➢   Note that, vec1[i] and vec1.at(i) are similar to access elements in vector. However, vec1.at(i) provides range checking but vec1[i] does not.

National Chiao Tung University

Laboratory Manual 13

Department of Electrical and Computer Engineering

June 11, 2012

Computer Intelligence on Automation(C.I.A.) Lab

Prof. Hung-Pin(Charles) Wen

✓ A vector of class objects can be created if the class has a default constructor.

```cpp
// lab13-1-2.cpp
#include <iostream>
#include <vector>
using std::cout;      using std::endl;
using std::vector;   using std::ostream;

class Point2D
{
private:
    int x;
    int y;
public:
    Point2D(): x(0), y(0){}
    Point2D(int a, int b): x(a), y(b){}
    friend ostream &operator << (ostream &out, const Point2D &p)
    {
        out << "(" << p.x << "," << p.y << ")";
    }
};

int main()
{
    int n = 10;
    vector<Point2D> vec(n);            // call Point2D()

    for (int i=0;i<vec.size();i++)
        vec[i] = Point2D(i*2,i*3);     // call Point2D(int a, int b)

    for (int i=0;i<vec.size();i++)
        cout << vec[i] << " ";
    cout << endl;

    return 0;
}
```

✓ Here demonstrates more operations supported by vector.

```cpp
// lab13-1-3.cpp
#include <iostream>
#include <vector>
using std::cout;      using std::endl;
using std::vector;

int main()
{
    int n = 5;
    vector<int> vec(n,-1);                 // vec = {-1,-1,-1,-1,-1}
    vector<int> u(3);
```

National Chiao Tung University
Laboratory Manual 13
Department of Electrical and Computer Engineering
June 11, 2012
Computer Intelligence on Automation(C.I.A.) Lab
Prof. Hung-Pin(Charles) Wen

```
    for (int i=0;i<3;i++) u[i] = i;        // u = {1,2,3}

    vec.insert(vec.begin()+2,u.begin(),u.end());
                                           // vec = {-1,-1,0,1,2,-1,-1,-1}
    vec.insert(vec.begin()+1,10);        // vec = {-1,10,-1,0,1,2,-1,-1,-1}
    vec.pop_back();                        // vec = {-1,10,-1,0,1,2,-1,-1}
    vec.erase(vec.begin()+3);            // vec = {-1,10,-1,1,2,-1,-1}
    vec.clear();                           // vec = {}

    for (int i=0;i<vec.size();i++)
        cout << vec[i] << " ";
    cout << endl;

    return 0;
}
```

➢ The functions begin() and end() return iterators to the first element and one-past-the-last element, respectively. It denotes the interval [begin,end).

➢ vec.insert(p,x) is used to add element x at position p and vec.insert(p,first,last) can insert a sequence [first,last) to position p.

➢ vec.erase(p) remove the element at position p. vec.clear() remove all elements.

✓ In <algorithm>, sort() is defined to sort the elements in increasing order. reverse() can reverse the elements in container and find() is used to find the specific element.

```
// lab13-1-4.cpp
#include <iostream>
#include <algorithm>
#include <vector>
using std::cout;       using std::endl;
using std::vector;     using std::ostream;

int main()
{
    int n = 10;
    vector<int> vec(n);

    for (int i=0;i<vec.size();i++)         // vec = {1,7,4,0,9,4,8,8,2,4}
        vec[i] = rand()%n;

    sort(vec.begin(), vec.end());          // vec = {0,1,2,4,4,4,7,8,8,9}
    reverse(vec.begin(), vec.end());       // vec = {9,8,8,7,4,4,4,2,1,0}

    for (int i=0;i<vec.size();i++)
        cout << vec[i] << " ";
    cout << endl;

    vector<int>::iterator iter = find(vec.begin(),vec.end(),8);
```

National Chiao Tung University
Department of Electrical and Computer Engineering
Computer Intelligence on Automation(C.I.A.) Lab

Laboratory Manual 13
June 11, 2012
Prof. Hung-Pin(Charles) Wen

```
        if (iter!=vec.end())
            cout << "8 is in the vector" << endl;
        else
            cout << "8 is not in the vector" << endl;



        return 0;
}
```

> The function sort() please elements of the vector in increasing order based on a less-than operation < by default.

TASK 13-2 MAP

✓ A map is a container whose elements are pairs of a key and a value. When indexed by the key, a map returns the corresponding value.

```
// lab13-2-1.cpp
#include <iostream>
#include <map>
#include <string>
using std::cout;     using std::endl;
using std::map;       using std::string;

int main()
{
    map<int,string> classroom;

    classroom[9912345] = "Jacky";
    classroom[9923456] = "John";
    classroom[9934567] = "Mary";

    for (map<int,string>::const_iterator iter = classroom.begin();
        iter != classroom.end(); iter++)
    {
        cout << "ID: " << iter->first << " ";
        cout << "name: " << iter->second << endl;
    }

    return 0;
}
```

✓ Here is another example to use map

```
// lab13-2-2.cpp
#include <iostream>
#include <map>
#include <string>
using std::cout;     using std::endl;
```

National Chiao Tung University
Laboratory Manual 13
Department of Electrical and Computer Engineering
June 11, 2012
Computer Intelligence on Automation(C.I.A.) Lab
Prof. Hung-Pin(Charles) Wen

```
using std::map;      using std::string;

int main()
{
    map<string,int> age;

    age["Mary"] = 22;
    age["Jacky"] = 18;
    age["John"] = 20;

    for (map<string,int>::const_iterator iter = age.begin();
        iter != age.end(); iter++)
    {
        cout << "name: " << iter->first << " ";
        cout << "age: " << iter->second << endl;
    }

    return 0;
}
```

➢ Note than map stores elements in increasing order based on a less-than operation <

TASK 13-3 EXERCISE

1. VECTOR

✓ Create a vector of Complex numbers and sort them by using the standard algorithm sort() in the order of decreasing absolute values. Note that you should define a Complex class.

✓ The output of the program should like as,

```
> ./ex13-1
Enter n: 5
Original sequence:
(1.6,4.7) (1.6,4.5) (7.5,8.4) (6.4,6.9) (3.8,3.4)
Sorted sequence:
(7.5,8.4) (6.4,6.9) (3.8,3.4) (1.6,4.7) (1.6,4.5)
```

2. MAP

✓ Rewrite the evaluation system defined in exercise 10-2.

✓ The input file performance.txt is defined as,

```
5
991003 Kevin 60 95 100 80 90 0 1 1 0 0
991001 Tom 50 60 80 90 70 1 0 0 1 1
991004 John 100 80 95 75 80 0 0 0 0 1
991005 Marry 55 60 70 90 65 1 1 0 0 0
991002 Jean 100 90 85 70 65 0 0 0 1 1
```

✓ The output of the program should like as,

```
> ./ex13-2 performance.txt
ID: 991001 name: Tom score: 85
ID: 991002 name: Jean score: 92
ID: 991003 name: Kevin score: 95
ID: 991004 name: John score: 91
ID: 991005 name: Marry score: 78
```

✓ The main function of the program is shown as follows,

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <map>
using namespace std;

class student
{
  …
};

int main(int argc, char *argv[])
{
    ifstream infile(argv[1],ios::in);

    map<int, student> classroom;

    // read file and store student's information

    for (map<int,student>::const_iterator iter = classroom.begin();
        iter != classroom.end(); iter++)
    {
        cout << "ID: " << …
        cout << "name: " << …
        cout << "score: " << …
    }

    return 0;
}
```