

UEE1303 S18: Object-Oriented Programming

LAB #9: STANDARD LIBRARIES ON CONTAINERS AND ALGORITHMS



What you will learn from Lab 9

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

LAB 9-1: 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 $n-1$.

```
// lab9-1-1.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;
}
```

1. Please fix the compiler error here.
2. 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.

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

```
// lab9-1-2.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;
}
```

✓ More operations are supported by vector.

```
// lab9-1-3.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);
    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}
    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;
}
```

1. The functions `begin()` and `end()` return iterators to the first element and one-past-the-last element, respectively. It denotes the interval `[begin,end)`.
2. `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`.
3. `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.

```
// lab9-1-4.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);
    // vec = {1,7,4,0,9,4,8,8,2,4}
    for (int i = 0; i < vec.size(); i++)
        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 = {9,8,8,7,4,4,2,1,0}
    for (int i = 0; i < vec.size(); i++)
        cout << vec[i] << " ";
    cout << endl;
    vector<int>::iterator iter
        = find(vec.begin(),vec.end(),8);
    if ( iter != vec.end() )
        cout << "8 is in the vector" << endl;
    else
        cout << "8 is not in the vector" << endl;
    return 0;
}
```

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

LAB 9-2: 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.

```
// lab9-2.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.

LAB 9-3: MAP

- ✓ You can also define a class template by adding prefix template<class T>.

```
// lab9-3.cpp
#include <iostream>
#include <map>
#include <string>
using std::cout; using std::endl;
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;
}
```

1. Note that map stores elements in increasing order based on a less-than operation <

EXERCISE 9-1: Sorting Complex Number

- ✓ 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.
- ✓ Please finish the code which is shown as follow,

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

class Complex{
    ...
};
bool operator < (const Complex& a,const Complex& b) {
    ...
}
void ReadFile(ifstream& infile, vector<Complex>& vec) {
    ...
}
int main(int argc, char** argv) {
```

```
if(argc != 2) return 0;
ifstream infile(argv[1]);
if(!infile.is_open()) {
    cout << "can not open file " << endl;
    return 0;
}
vector<Complex> vec;
ReadFile(infile, vec);

cout << "Original sequence:" << endl;
for(size_t i = 0; i < vec.size(); i++)
    cout << vec[i] << " ";
cout << endl;

sort(vec.begin(),vec.end());

cout << "Sorted sequence:" << endl;
for(size_t i = 0; i < vec.size(); i++)
    cout << vec[i] << " ";
cout << endl;
return 0;
}
```

- ✓ The output of the program should like as,

```
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)
```

EXERCISE 9-2: Book Store

- ✓ Write a book store system that includes four functions
1. Show all information of book store (ID, Name, and Number)
 2. Add new books to book store system
 3. Sell books
 4. Exit the system

- ✓ The output of the program should like as,

```
> ./ex9-2
1. show information of book store
2. add new books
3. sell books
4. exit
1
  ID              Name      Number
  1          C++ Programing    1
  2              Java          2
  3             Python          3
1. show information of book store
2. add new books
3. sell books
4. exit
2
enter Name and Number of new book
Parallel Programming
```

```
5
1. show information of book store
2. add new books
3. sell books
4. exit
1
  ID          Name          Number
  1          C++ Programing    1
  2           Java            2
  3          Python           3
  4    Parallel Programing     5
1. show information of book store
2. add new books
3. sell books
4. exit
3
enter sell id
1
enter sell number
1
1. show information of book store
2. add new books
3. sell books
4. exit
1
  ID          Name          Number
  1          C++ Programing    0
  2           Java            2
  3          Python           3
  4    Parallel Programing     5
1. show information of book store
2. add new books
3. sell books
4. exit
4
```

✓ The example of Book is shown below,

```
class Book {
    string Name; // name of book
    int Number; // how many books in your store

    public:
    Book(){};
    Book(string _name, int _number):Name(_name),Number(_number){}
    ...
};

int main() {
    map<int, Book> BookMap;
    BookMap[1] = Book("C++ Programing", 1);
    BookMap[2] = Book("Java", 2);
    BookMap[3] = Book("Python", 3);
    int step;
    while(1) {
        cout <<"1. show information of book store" << endl;
        cout <<"2. add new books" << endl;
```

```
        cout <<"3. sell books " << endl;
        cout <<"4. exit" << endl;
        cin >> step;
        if(step==1) {
            ...
        }
        else if(step==2) {
            cout <<"enter Name and Number of new book" << endl;
            ...
        }
        else if(step==3) {
            cout << "enter sell id" << endl;
            ...
        }
        else if(step==4) {
            break;
        }
    }
    return 0;
}
```